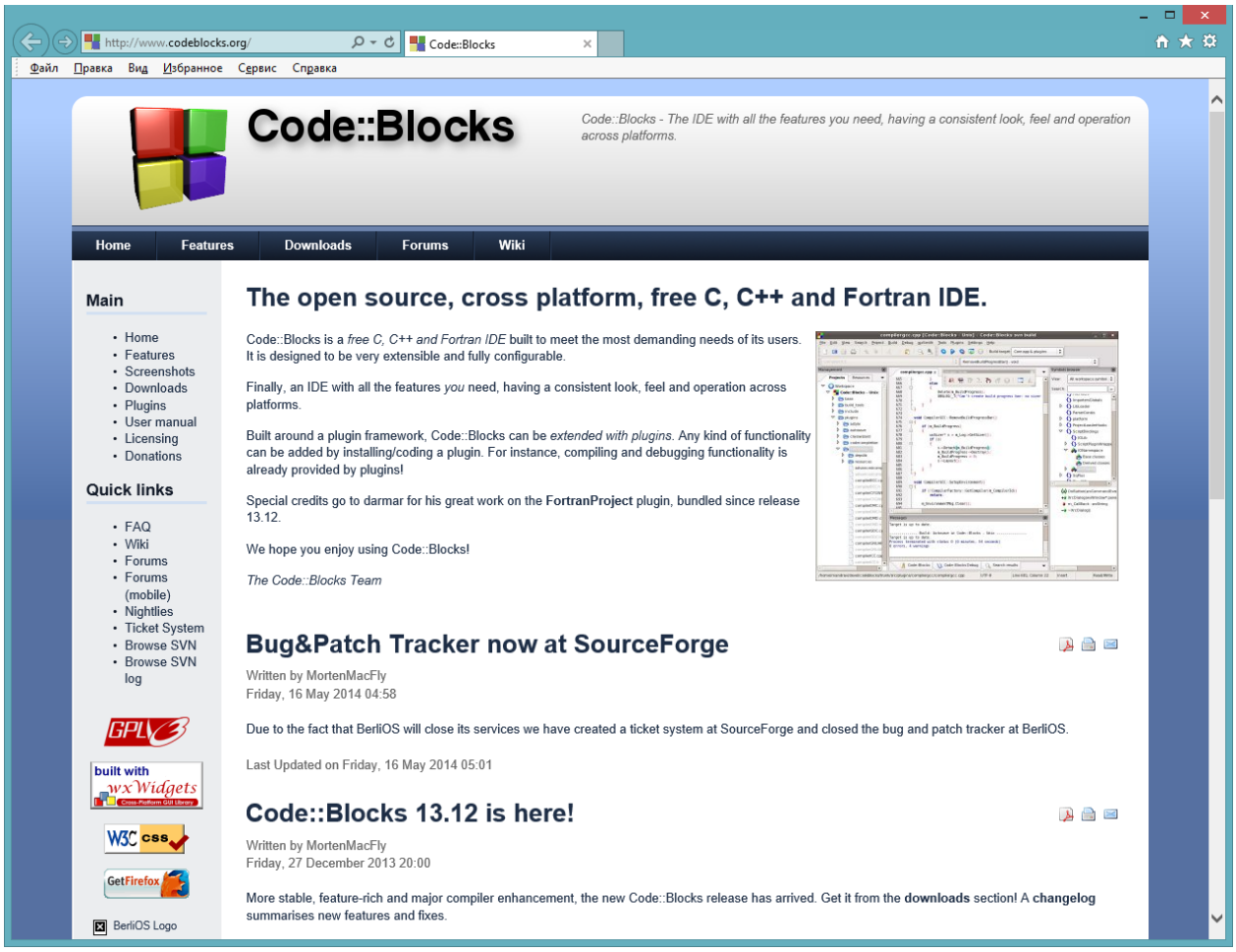
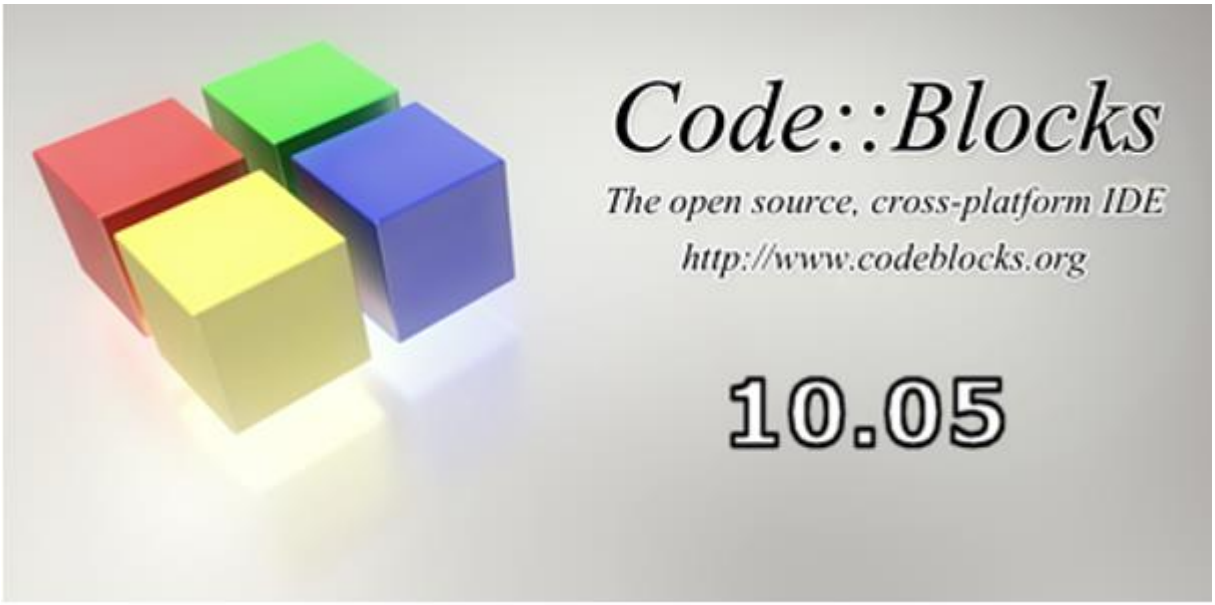
В.Н. Гололобов

О ПЕРЕВОДЕ ОДНОГО РУКОВОДСТВА

Часть 2. Code::Blocks



Москва -2015



CodeBlocks

Руководство

Version 1.1

Спасибо группе CodeBlocks:

Anders F. Bjo¨rklund (afb), Biplab Kumar Modak (biplab), Bartomiej wiecki (byo), Paul A. Jimenez (ceniza), Koa Chong Gee (cyberkoa), Daniel Orb (daniel2000), Lieven de Cock (killerbot), Yiannis Mandravellos (mandrav), Mispunt (mispunt), Martin Halle (mortenmacfly), Jens Lody (jens), Jerome Antoine (dje), Damien Moore (dmoore), Pecan Heber (pecan), Ricardo Garcia (rickg22), Thomas Denk (thomasdenk), tiwag (tiwag)

1 Управление проектом

Инструкции главы 3 на странице 53 и ?? на странице ?? – официальная документация CodeBlocks Wiki сайта и доступна только на английском.

Иллюстрация ниже показывает вид пользовательского интерфейса CodeBlocks.

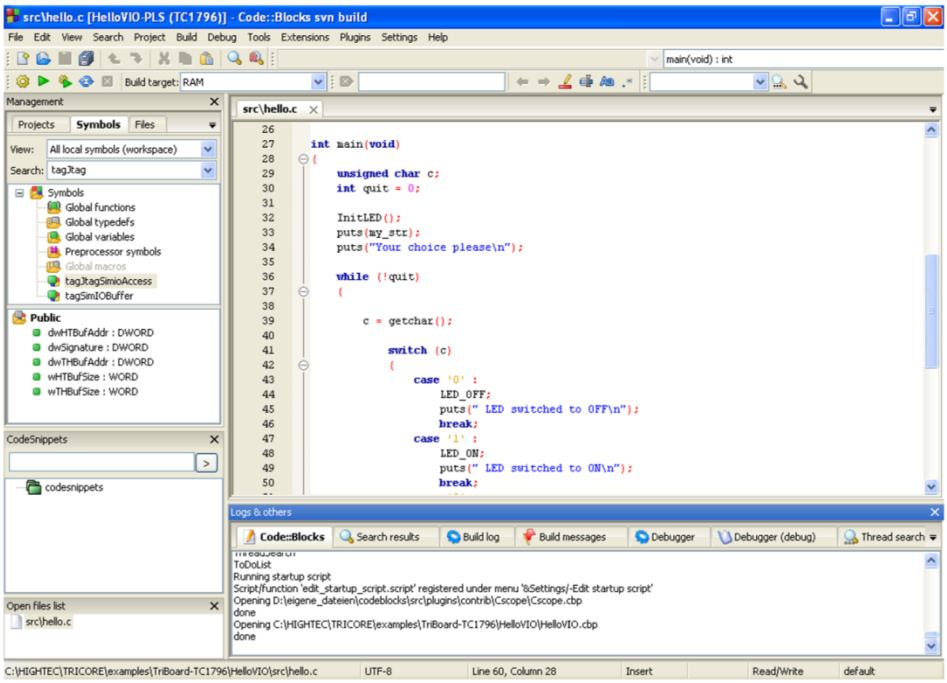


Рис. 1.1. IDE CodeBlocks

Management

Это окно содержит интерфейс «Projects», на который в дальнейшем будем ссылаться как на обозреватель проекта. Этот обозреватель показывает все проекты, открытые в CodeBlocks в настоящий момент. Закладка «Symbols» окна Management показывает символы, переменные и т.д.

Editor

На иллюстрации выше исходный текст, названный hello.c, открыт с подсветкой синтаксиса в редакторе.

Open files list

Показывает список всех файлов, открытых в редакторе, в этом примере: hello.c.

CodeSnippets

Может отображаться с помощью меню ’View’ →’CodeSnippets’. Здесь вы можете управлять модулями текста, указывая на файл или на url.

Logs & others

Это окно используется для вывода результатов поиска, сообщений компилятора и т.д.

Панель состояния даёт обзор следующих параметров:

- Абсолютный путь к открытому в редакторе файлу.
- Редактор использует предопределённую кодировку символов операционной системы. Эти настройки будут отображаться с *default*.
- Номера строк и колонок текущей позиции курсора в редакторе.
- Конфигурация режима работы клавиатуры для вставки текста (Insert или Overwrite).
- Текущее состояние файла. Модифицированный файл будет отмечен как *Modified*, иначе это поле останется пустым.
- Разрешение на работу с файлом. Файл с меткой read only (только для чтения) будет помечен как *Read only* в панели состояния. В окне 'Open files list' эти файлы будут подчёркнуты иконкой с закрытым замком.

  Примечание:

  В активном редакторе пользователь может выбрать раздел свойства контекстного меню. В появившемся диалоге на закладке 'General' опция 'File is readonly' может быть установлена. Эта установка опции приведёт к доступу только для чтения соответствующего файла в CodeBlocks, но оригинальные атрибуты чтения и записи файла в файловой системе модифицированы не будут.

- Если вы запускаете CodeBlocks с опцией --personality=<profile> в командной строке, тогда панель состояния отобразит текущий профиль пользователя, иначе будет показано *default*. Установки CodeBlocks сохраняются в соответствующем файле конфигурации <personality>.conf.

CodeBlocks предполагает очень гибкое и полное управление проектом. Следующий текст затрагивает только некоторые из возможностей управления проектом.

1.1 Обозреватель проекта

В CodeBlocks исходники и настройки для сборки хранятся в файле проекта <name>.cbp. C/C++ исходные и соответствующие заголовочные файлы являются типичными компонентами проекта. Самый лёгкий путь создать новый проект – это выполнить команду 'File' →'Project' и выбрать тип проекта в помощнике. И вы можете добавить файлы в проект через контекстное меню 'Add files' в окне Management.

CodeBlocks представляет проект файлами, отсортированными по категориям, согласно с расширениями файлов. Вот предустановленные категории:

Sources Включает исходные файлы с расширениями *.c;*.cpp;.

ASM Sources Включает исходные файлы с расширениями *.s;*.S;*.ss;*.asm.

Headers Включает, в том числе, файлы с расширением *.h;.

Resources Включает файлы описания вида для окон wxWidgets с расширениями *.res;*.xrc;. Эти типы файлов показаны на закладке 'Resources' окна Manangement.

Настройки для типов и категорий файлов могут быть подстроены с помощью контекстного меню 'Project tree' →'Edit file types & categories'. Здесь вы можете также определить пользовательские категории для ваших собственных расширений файлов. Например, если вы хотите включить скрипты компоновщика с расширением *.ld в категорию, названную Linkerscript, вам достаточно создать новую категорию.

> Примечание:
>
> Если вы деактивируете 'Project tree' →'Categorize by file types' в контекстном меню, отображение категории будет выключено, а файлы будут перечислены так, как они хранятся в файловой системе.

1.2 Примечания к проекту

В CodeBlocks так называемые примечания могут сохраняться для проекта. Эти примечания будут содержать краткие описания или подсказки для соответствующего проекта. Отображение этой информации в процессе открывания проекта поможет другим пользователем быстрее познакомиться с проектом. Показ примечаний может быть включён и выключен на закладке 'Notes' раздела 'Properties' проекта.

1.3 Шаблоны проекта

CodeBlocks поставляется с различными шаблонами проектов, которые отображаются, когда создаётся новый проект. Однако возможно сохранять и пользовательские шаблоны для подборки ваших собственных спецификаций в шаблонах для переключения компиляторов, оптимизации, машинно-зависимых переключений и т.д. Эти шаблоны будут храниться в Documents and Settings\<user>\Application Data\codeblocks\UserTemplates директории. Если шаблоны предназначены для всех пользователей, они должны копироваться в соответствующие директории установки CodeBlocks. Эти шаблоны будут затем отображаться при следующих запусках CodeBlocks под 'New' →' Project' →'User templates'.

> Примечание:
>
> Доступные шаблоны в Project Wizard могут быть отредактированы выбором с помощью щелчка правой клавиши мышки.

1.4 Создание проектов из целей сборки

В проектах необходимо иметь разные варианты доступности проекта. Эти варианты называются цели сборки (Build Targets). Они отличаются опциями компилятора, отладочной информацией и/или выбором файлов. Цель сборки может также быть передана в ведение отдельного проекта. Чтобы это сделать, щёлкните по 'Project' →'Properties', выберите вариант на закладке 'Build Targets' и щёлкните по кнопке **Create project from target** (см. рис. 1.2 на стр. 5).

1.5 Виртуальные цели

Проекты могут быть в дальнейшем структурированы в так называемые виртуальные цели (Virtual Targets). Структура часто используемого проекта состоит из двух целей сборки - одна 'Debug', которая состоит из отладочной информации, а другая 'Release' без этой информации.

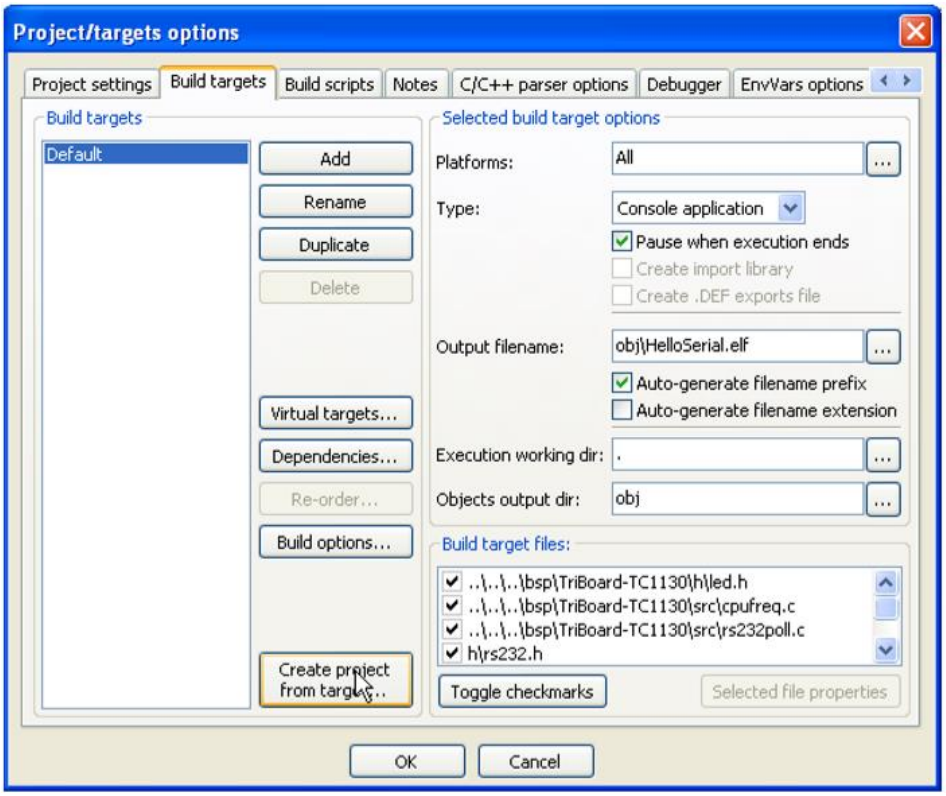


Рис. 1.2. Цели сборки

Добавлением Virtual Targets через 'Project' →'Properties' →'Build Targets' индивидуальные цели сборки можно комбинировать. Например, виртуальная цель 'All' может создать цели Debug и Release одновременно. Виртуальные цели показаны в окне символов (symbol bar) компилятора под Build Targets.

1.6 Пред- и пост-сборочные шаги

CodeBlocks позволяет выполнить ряд операций до и после компиляции проекта. Эти операции называются Prebuilt или Postbuilt Steps. Типичные Postbuilt Steps это:

- Создание hex-файла из завершённого объекта.
- Манипуляции с объектами с помощью objcopy.
- Генерация дампа файлов с помощью objdump.

Пример

Дизассемблирование из объектных файлов (object) под Windows. Непосредственный вызов требуемого файла опцией **cmd /c**.

```
cmd /c objdump -D name.elf > name.dis
```

Архивирование проекта может быть другим примером пост-сборочного шага. Для этого создайте цель сборки 'Archive' и включите следующие инструкции в Postbuilt Step:

```
zip -j9 $(PROJECT_NAME)_$(TODAY).zip src h obj $(PROJECT_NAME).cbp
```

Этой командой активный проект и его исходники, заголовочный файл и объектные файлы будут упакованы в файл zip-формата. При этом встроенные переменные **$(PROJECT_NAME)** и **$(TODAY),**

имя проекта и текущая дата, будут извлечены (см. раздел 3.2 на стр. 54). После выполнения цели 'Archive' сжатый файл будет сохранён в директории проекта.

В директории share/codeblocks/scripts вы найдёте некоторые примеры скриптов. Вы можете добавить скрипт через меню 'Settings' →'Scripting' и зарегистрировать в меню. Если вы выполняете этот скрипт, выполните dist из меню, тогда все файлы, принадлежащие проекту, будут сжаты в архив <project>.tar.gz.

1.7 Добавление скриптов в цель сборки

CodeBlocks позволяет использовать команды меню в скриптах. Скрипт (сценарий) представляет ещё одну степень свободы при управлении генерацией вашего проекта.

> Примечание:
>
> Скрипт также может быть включён в цель сборки.

1.8 Рабочее пространство и зависимости проекта

В CodeBlocks можно открыть несколько проектов. Сохранив открытые проекты через 'File' →'Save workspace', вы можете собрать их в едином рабочем пространстве (workspace) под <name>.workspace. Если вы открываете <name>.workspace при следующем запуске CodeBlocks, все проекты будут вновь показаны.

Сложные программные системы состоят из компонентов, которые обработаны в разных проектах CodeBlocks. Кроме того, при создании подобных программных систем часто возникает зависимость между этими проектами.

Пример

Проект A содержит фундаментальные функции, которые доступны для других проектов в форме библиотеки. Теперь, если исходники этого проекта модифицируются, тогда библиотека должна быть пересобрана. Для сохранения согласованности между проектом B, который использует функции, и проектом A, который реализует функции, проект B должен зависеть от проекта A. Необходимая информация о зависимости проектов хранится в соответствующем рабочем пространстве, так что каждый проект может создаваться отдельно. Использование зависимостей делает возможным управлять порядком, в котором проекты будут сгенерированы. Зависимости проектов могут быть заданы с помощью выбора в меню 'Project' →'Properties', а затем щелчком по кнопке **Project's dependencies**.

1.9 Включение ассемблерных файлов

В окне Management обозревателя проекта (Project View), ассемблерные файлы показаны в категории ASM Sources. Пользователь может изменить перечисление этих файлов в категориях (см. раздел 1.1 на стр. 3). Щелчок правой клавишей мышки по одному из ассемблерных файлов откроет контекстное меню. Выберите 'Properties', что откроет новое окно. Теперь выберите закладку 'Build' и активируйте два поля 'Compile file' и 'Link file'. Затем выберите закладку 'Advanced' и выполните следующие шаги:

1. Установите 'Compiler variable' в CC.
2. Выберите компилятор под 'For this compiler'.
3. Выберите 'Use custom command to build this file'.

4. В окне введите:

```
$compiler $options $includes <asopts> -c $file -o $object
```

Переменные CodeBlocks маркируются с помощью $ (см. раздел 3.4 на стр. 58). Они устанавливаются автоматически, так что вам остаётся только заменить ассемблерную опцию <asopt> вашей собственной.

1.10 Редактор и инструменты

1.10.1 Предопределённый код

Правила кодирования компаний требуют, чтобы исходные файлы имели стандартный вид. CodeBlocks делает возможным включение предопределённого содержания в начале файла автоматически при создании новых C/C++ исходных и заголовочных файлов. Это предопределённое содержание называется предопределённым кодом. Эти установки можно выбрать под 'Stettings'→'Editor' Default Code. Если вы создали новый файл, тогда выполняется макро расширение переменных, определённых с помощью меню 'Settings' →'Global variables'. Новый файл может создаваться с помощью меню 'File' →'New' →'File'.

Пример

```
/*************************************************************************
*       Project: $(proejct)
*       Function:
*************************************************************************
*       $Author: mario $
*       $Name: $
************************************************************************* *
*       Copyright 2007 by company name
* ************************************************************************/
```

1.10.2 Аббревиатура

Значительная часть вводимого текста может храниться в CodeBlocks в виде предопределённой аббревиатуры. Это делается с помощью выбора 'Settings' →'Editor' и определения аббревиатуры под именем <name>, которое затем может вызываться горячими клавишами с клавиатуры **Ctrl-J** (см. рис. 1.3 на стр. 8).

Также возможна параметризация включением переменных $(NAME) в аббревиатуры.

```
#ifndef $(Guard token)
#define $(Guard token)
#endif // $(Guard token)
```

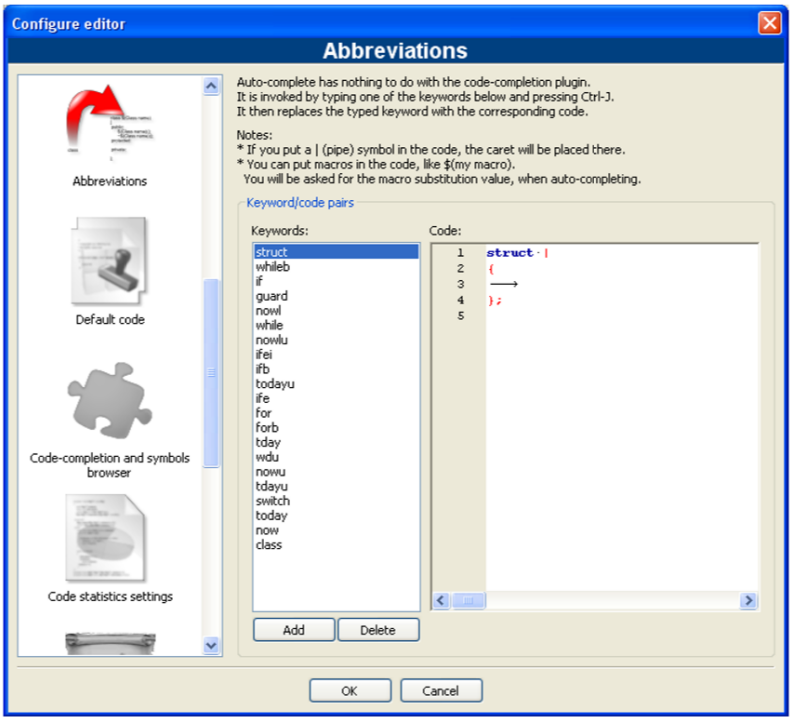


Рис. 1.3. Определение аббревиатур

Когда выполняется аббревиатура <name> в исходном тексте и выполняется **Ctrl-J**, содержимое переменной запрашивается и вставляется.

1.10.3 Персонализация

Установки CodeBlocks сохраняются в данных приложения в файле, названном <user>.conf в директории codeblocks. Этот файл конфигурации содержит информацию из ряда: последний открытый проект, настройки редактора, отображение в окне символов и т.д. По умолчанию персонализация 'default' настроена так, что конфигурация хранится в файле default.conf. Если CodeBlocks вызывается из командной строки с параметром --personality=myuser, установки сохранятся в файле myuser.conf. Если профиль пока не существует, он будет автоматически создан. Эта процедура делает возможным создать соответствующие профили для разной по характеру работы. Если запускать CodeBlocks из командной строки с дополнительным параметром --personality=ask, отобразится окно выбора со всеми доступными профилями.

> Примечание:
>
> Имя текущего профиля/персонализации отображается в правом углу панели состояния.

1.10.4 Файлы конфигурации

Установки CodeBlocks хранятся в профиле default.conf в директории codeblocks вашей директории Application Data. При использовании персонализаций (см. подраздел 1.10.3 на стр. 8) детали конфигурации будут храниться в файле <personality>.conf.

Инструмент cb_share_conf, который можно найти в директории установки CodeBlocks, используется для обслуживания и запоминания этих настроек.

Если вы хотите определить стандартные установки для нескольких пользователей компьютера, файл конфигурации default.conf следует скопировать в директорию \Documents and Settings\Default User\Application Data\codeblocks. При первом запуске CodeBlocks скопирует предустановки из 'Default User' в директорию Application Data текущего пользователя.

Чтобы создать портативную версию CodeBlocks на USB флэшке, сделайте следующее. Скопируйте установку CodeBlocks на флэшку и сохраните конфигурационный файл default.conf в этой директории. Конфигурация будет использоваться как глобальные установки. Пожалуйста, позаботьтесь, чтобы файл был перезаписываем, иначе изменения конфигурации не будут сохраняться.

1.10.5 Навигация и поиск

В CodeBlocks есть разные способы быстрой навигации по файлам и функциям. Типичная процедура – закладки. С помощью горячих клавиш **Ctrl-B** закладки устанавливаются и удаляются в исходном файле. С помощью **Alt-PgUp** вы можете перемещаться к предыдущей закладке, а с помощью **Alt-PgDn** к следующей.

Если вы выбрали рабочее пространство или проект в рабочем поле обозревателя проектов, вы сможете поискать файл в проекте. Достаточно выбрать 'Find file' из контекстного меню, затем ввести имя файла и файл будет отмечен. Если вы нажмёте **Enter**, этот файл будет открыт в редакторе (см. Рис. 1.4 на стр. 9).

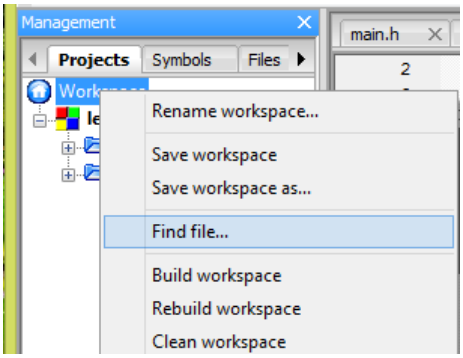


Рис. 1.4. Поиск файлов

В CodeBlocks вы можете легко перемещаться между заголовочными/исходными файлами:

1. Установите курсор в месте, где заголовочный файл включается (#include) и откройте этот файл с помощью контекстного меню 'open include file' (см. рис. 1.5 на стр. 10).
2. Переключение между заголовочным и исходным файлом через контекстное меню 'Swap header/source'.
3. Выберите декларацию (#define) в редакторе и выберите 'Find declaration' из контекстного меню, чтобы открыть файл с его объявлением.

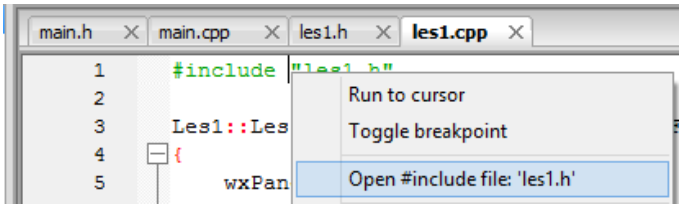


Рис. 1.5. Открывание заголовочного файла

CodeBlocks предлагает несколько способов поиска в файле или директории. Диалоговое окно поиска открывается с помощью 'Search' →'Find' (**Ctrl-F**) или 'Find in Files' (**Ctrl-Shift-F**). **Alt-G** и **Ctrl-Alt-G** – другие полезные функции. Диалог, который будет открыт этими горячими клавишами, позволит вам выбрать файлы/функции, а затем перейти к реализации выбранных функций (см. рис. 1.6 на стр. 10) или открыть файл в редакторе. Вы можете использовать подстановочные знаки ∗ или ? в диалоге для последовательного поиска.

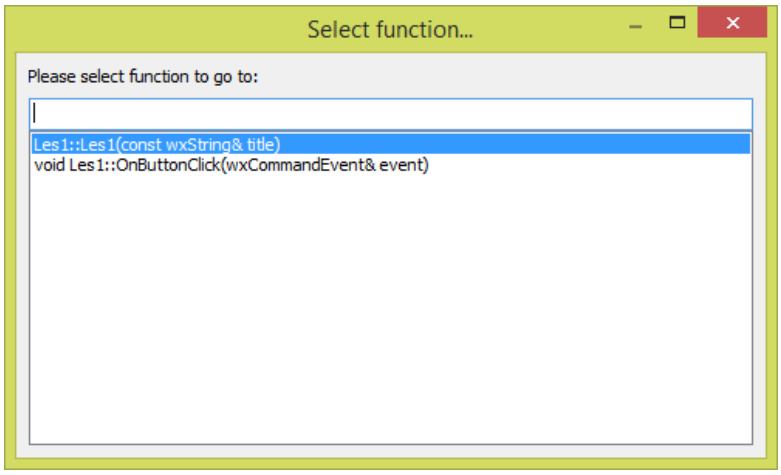


Рис. 1.6. Поиск функций

Примечание:

Горячие клавиши **Ctrl-PgUp** помогут вам перейти к предыдущей функции, а **Ctrl-PgDn** к следующей.

В редакторе вы можете открыть новый диалог *Open Files* с помощью **Ctrl-Tab** и переключаться между перечисленными пунктами. Если нажата клавиша **Ctrl**, тогда файл может выбираться разными способами:

1. Если вы выделили файл левой клавишей мышки, он будет открыт.
2. Когда вы нажимаете клавишу **Tab**, вы будете переключаться между перечисленными пунктами. Отпустив клавишу **Crtl**, вы откроете выбранный файл.
3. Если вы перемещаете курсор мышки по перечисленным пунктам, тогда текущее выделение будет подсвечено. Отпустив клавишу **Crtl**, вы откроете выделенный файл.
4. Если указатель мышки вне подсвеченного выделения, тогда вы можете использовать колёсико мышки для переключения между пунктами. Отпустив клавишу **Crtl**, вы откроете выбранный файл.

Общая процедура при разработке программ – это продираться через множество функций, которые реализованы в разных файлах. Плагин Browse Tracker поможет вам решить эту проблему, показав вам последовательность выбора файлов. Вы можете затем комфортно перемещаться по вызовам функций (см. раздел 2.8 на стр. 39). Отображение номеров строк в CodeBlocks может быть активировано с помощью 'Settings' →'General Settings' в поле 'Show line numbers'. Горячие клавиши **Ctrl-G** или команда меню 'Search' →'Goto line' помогут вам перемещаться по нужным строкам.

> Примечание:
>
> Если вы удерживаете клавишу **Ctrl**, а затем выделяете текст в редакторе CodeBlocks, вы можете выполнить поиск Google search с помощью контекстного меню.

1.10.6 Обозреватель символов

Окно Management в CodeBlocks показывает дерево символов C/C++ исходников для навигации по функциям или переменным. В качестве границ этого обзора вы можете задать текущий файл или проект, или всё рабочее пространство.

> Примечание:
>
> Результатом ввода термина или имён символов в маску ввода 'Search' в Symbol Browser будет отфильтрованный обзор символов, если совпадения обнаружатся.

Для символов существуют следующие категории:

Global functions Перечисляются реализации глобальных функций.

Global typedefs Перечисляется использование определений typedef.

Global variables Отображаются символы глобальных переменных.

Preprocessor symbols Перечисляются директивы препроцессора, созданные #define.

Global macros Перечисляются макросы директив препроцессора.

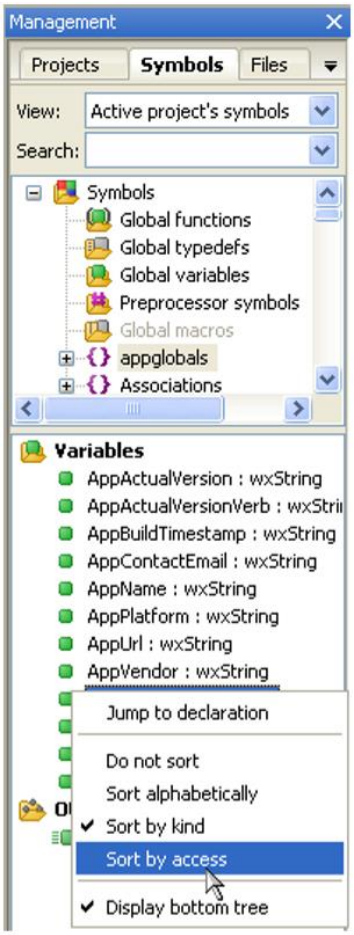


Рис. 1.7. Обозреватель символов

Структуры и классы отображаются в ’bottom tree’, а последовательность сортировки может модифицироваться с помощью контекстного меню. Если категория выбрана щелчком клавиши мышки, найденные символы отобразятся в нижней части окна (см. рис. 1.7 на стр. 12). Двойной щелчок по символу откроет файл, в котором определён символ или реализована функция, и осуществит переход к соответствующей строке. Автоматическое обновление обозревателя символов без сохранения файла может быть активировано через ’Settings’ →’Editor’ →’Code Completion’ в меню (см. рис. 1.8 на стр. 12). Для проектов с множеством символов будет затронута производительность CodeBlocks.

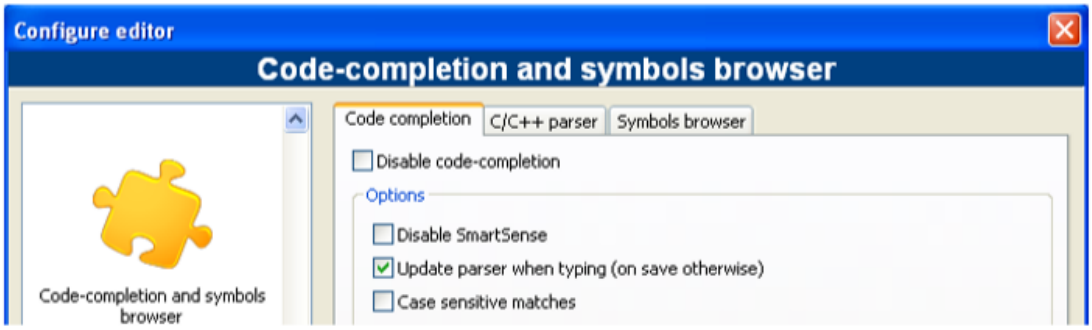


Рис. 1.8. Разрешение анализа в реальном времени

Примечание:

В редакторе список классов может отображаться с помощью контекстного меню ’Insert Class method declaration implementation’ или ’All class methods without implementation’.

1.10.7 Включение внешних файлов помощи

Окружение разработки CodeBlocks поддерживает включение внешних файлов помощи через меню ’Settings’ →’Environment’. Включите руководство по вашему выбору в chm формате в ’Help Files’, выберите ’this is the default help file’ (см. рис. 1.9 на стр. 13). Ввод $(keyword) – это заглушка для выбора объекта в ваш редактор. Теперь вы можете выделить функцию в открытом исходном файле в CodeBlocks щелчком мышки, а соответствующий документ появится при нажатии на клавишу **F1**.

Если вы включили множество файлов, вы можете выделить нужное в редакторе и выбрать файл подсказки из контекстного меню ’Locate in’ для CodeBlocks, чтобы осмотреть всё.

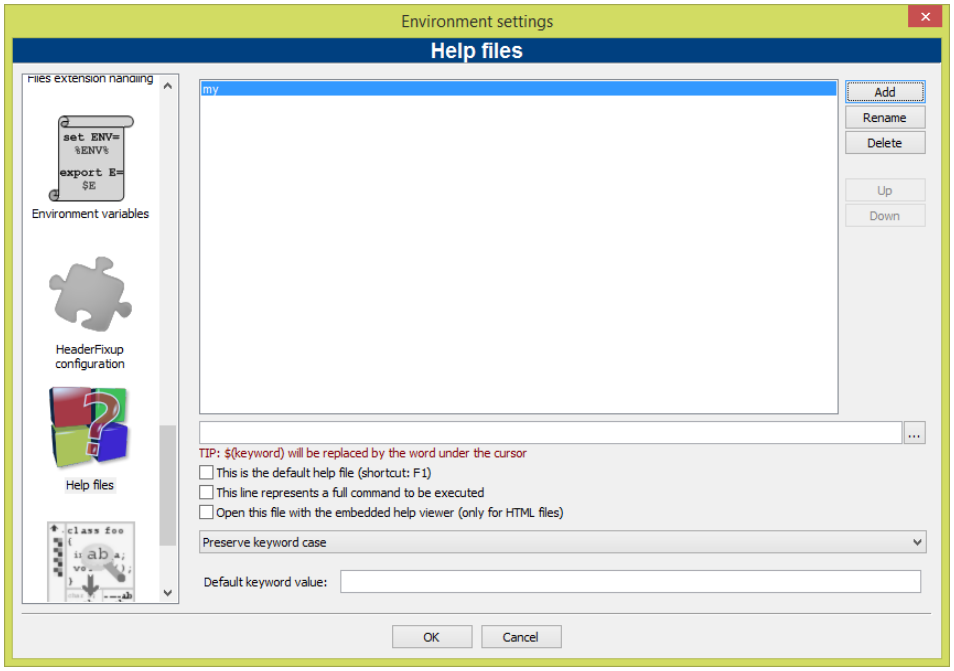


Рис. 1.9. Установки для файлов помощи

В CodeBlocks вы можете добавить даже поддержку man страниц. Только добавьте ввод ’man’ и задайте путь, как указано ниже.

```
man:/usr/share/man
```

CodeBlocks поддерживает ’Embedded HTML Viewer’, который может использоваться для отображения простых html файлов и поиска ключевых слов в этом файле. Достаточно сконфигурировать путь к html файлу, который нужен, и отметить разрешение ’Open this file with embedded help viewer’ с помощью меню ’Settings’ →’Environment’ →’Help Files’.

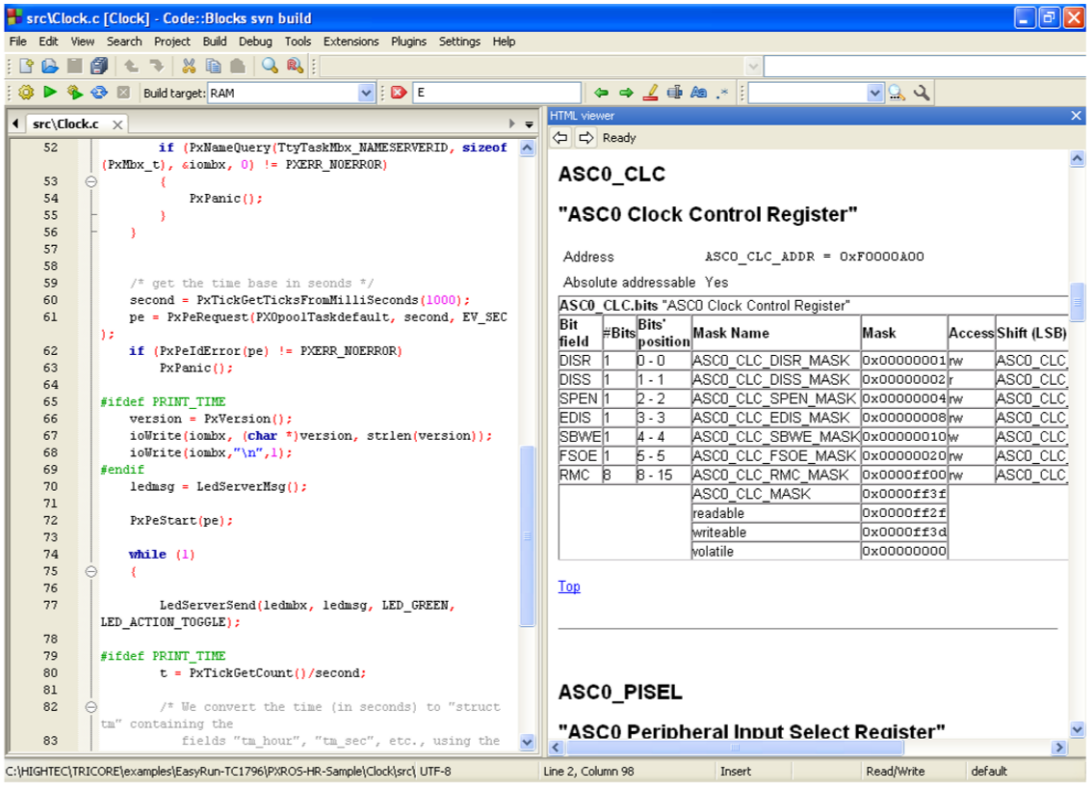


Рис. 1.10. Встроенный HTML обозреватель

Примечание:

Если вы выделили html файл двойным щелчком мышки в проводнике операционной системы (см. раздел 2.7 на стр. 35), тогда запустится встроенный обозреватель html, при условии, что нет ассоциированной с расширением html файлов поддержки.

1.10.8 Включение внешних инструментов

Включение внешних инструментов возможно в CodeBlocks через 'Tools' →'Configure Tools' →'Add' . Встроенные переменные (см. раздел 3.2 на стр. 54) для параметров инструмента могут также быть доступны. Более того, есть несколько видов опций для запуска внешних приложений. В зависимости от опций запускаемые внешние приложения будут остановлены, когда вы покидаете CodeBlocks. Если приложения останутся открыты после закрывания CodeBlocks, должна быть установлена опция 'Launch tool visible detached'.

1.11 Краткие указания по работе с CodeBlocks

В этом разделе мы покажем некоторые полезные настройки в CodeBlocks.

1.11.1 Отслеживание модификаций

CodeBlocks поддерживает возможность отслеживания модификаций в файле исходного кода, показывая полоску по краю, отмечающую изменение. Модификация маркируется жёлтой полоской, а модификация, которая уже сохранена, помечается зелёной полоской (см. рис. 1.11 на стр. 15). Вы можете перемещаться между вашими изменениями с помощью меню 'Search' →'Goto next changed line' или 'Search' →'Goto previous changed line'. Аналогичная функциональность доступна с помощью горячих клавиш **Ctrl-F3** и **Ctrl-Shift-F3**.

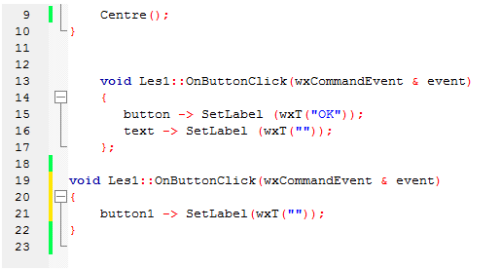


Рис. 1.11. Отслеживание модификаций

Эта возможность может разрешаться или запрещаться установкой флажка 'Use Changebar' в меню 'Settings' →'Editor' →'Margins and caret'.

> Примечание:
>
> Если модифицированный файл закрыт, тогда история изменений подобно undo/redo и changebars будет утрачена. С помощью меню 'Edit'→'Clear changes history' или соответствующего контекстного меню вы может очистить историю изменений даже в том случае, когда файл остаётся открыт.

1.11.2 Обмен данными с другими приложениями

Данными можно обмениваться между CodeBlocks и другими приложениями. Для этой коммуникации между процессами DDE (Dynamic Data Exchange, динамический обмен данными) для Windows и других операционных систем используется коммуникация на основе TCP.

С этим интерфейсом разные команды следующего синтаксиса могут быть посланы в образец CodeBlocks.

```
[<command>("<parameter>")]
```

Следующие команды доступны в настоящий момент:

Open Команда

```
[Open("d:\temp\test.txt")]
```

используует параметр, в нашем случае задаётся файл с абсолютным путём и открывается в существующем образце CodeBlocks или запускается новый образец, если это нужно.

OpenLine Эта команда октрывает файл на заданной строке в образце CodeBlocks. Номер строки задаётся с : line.

```
[OpenLine("d:\temp\test.txt:10")]
```

Raise Устанавливает фокус на образце CodeBlocks. Параметр не должен передаваться.

1.11.3 Конфигурирование переменных окружения

Конфигурация для операционной системы задаётся так называемыми переменными окружения. Переменная окружения PATH, например, задаёт путь к установленному компилятору.

Операционная система обслуживает эти переменные окружения от начала к концу, то есть, записанное в конце будет найдено последним. Если установлены разные версии компиляторов или других приложений, обнаружатся следующие ситуации:

- Вызывается неправильная версия программы.
- Установленные пакеты программы вызывают друг друга.

Таким образом, возможен случай, когда разные версии компиляторов или других инструментов становятся обязательны для разных проектов. Одна возможность в этом случае – это изменить переменные окружения в системе управления для каждого из проектов. Однако эта процедура подвержена ошибкам и не отличается гибкостью. Из этих соображений CodeBlocks предлагает элегантное решение. Разные конфигурации переменных окружения могут создаваться так, что будут использоваться только внутри CodeBlocks. Дополнительно вы можете переключаться между этими конфигурациями. Рисунок 1.12 на странице 16 показывает диалог, который вы можете открыть с помощью 'Environment Varibales' под 'Settings' →'Environment'. Конфигурация создаётся кнопкой **Create**.

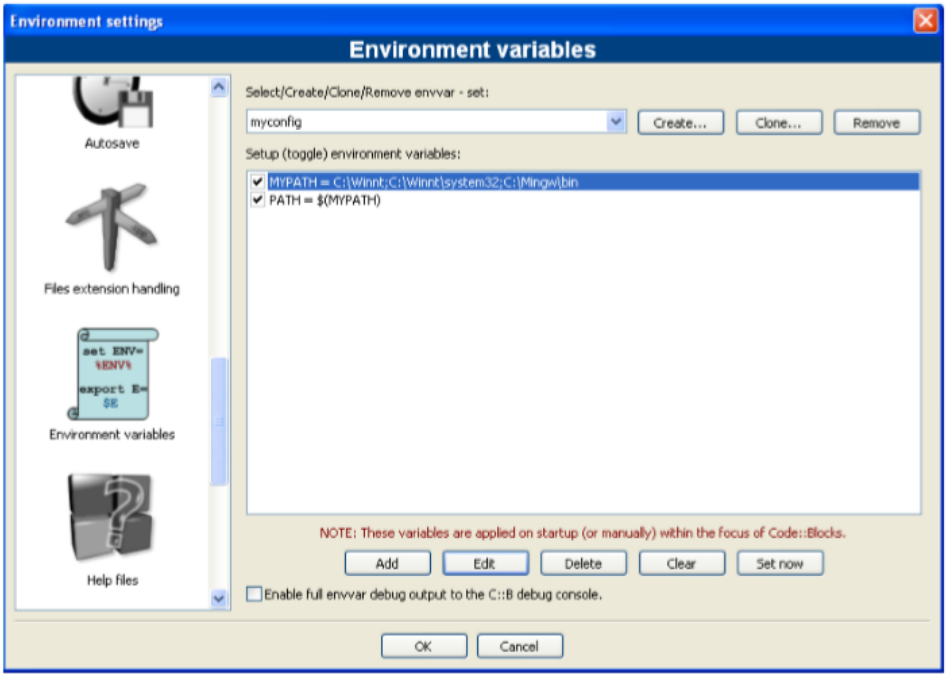


Рис. 1.12. Переменные окружения

Доступ и границы переменных окружения, созданных здесь, ограничен в CodeBlocks. Вы можете расширить эти переменные окружения подобно другим переменным CodeBlocks через $(NAME).

> Примечание:
>
> Конфигурация переменных окружения для каждого из проектов может быть выбрана из контекстного меню 'Properties' закладки 'EnvVars options'.

Пример

Вы можете записать используемое окружение в postbuild Step (см. раздел 1.6 на стр. 5) в файл <project>.env и хранить его в вашем проекте.

```
cmd /c echo \%PATH\% > project.env
```

или в Linux

```
echo \$PATH > project.env
```

1.11.4 Переключение между видами

В зависимости от поставленной задачи может оказаться полезным иметь разные конфигурации или виды в CodeBlocks, и хранить эти configurations/views. По умолчанию настройки (как show/hide, показать/скрыть окно символов, внешний вид и т.д.) хранятся в конфигурационном файле default.conf. Используя опцию командной строки --personality=ask при старте CodeBlocks, можно выбрать разные настройки. Отдельно от этих глобальных установок может сложиться ситуация, когда вы хотели бы переключаться между разными видами окон и обзорами символов во время сессии. Редактирование файлов и отладка проекта – типичные примеры подобной ситуации. CodeBlocks предлагает механизм для хранения и выбора разных видов, избавляя пользователя от необходимости часто открывать и закрывать окна или включать обзор символов вручную. Для сохранения видов выберите в меню 'View' →'Perspectives' →'Save current' и введите имя в <name>. Команда 'Settings' →'Editor' →'Keyboard shortcuts' →'View' →'Perspectives' →'<name>' позволяет определить горячие клавиши для этого процесса. Этот механизм делает возможным переключение между разными видами простым нажатием горячих клавиш.

> Примечание:
>
> Другой пример – редактирование файла в режиме полного экрана без окошка символов. Вы можете создать вид, такой как 'Full', и назначить горячие клавиши для этой цели.

1.11.5 Переключение между проектами

Если несколько проектов или файлов открыты одновременно, пользователю нужен способ быстро переключаться между ними. CodeBlocks имеет несколько горячих клавиш для этой ситуации.

Alt-F5	Активировать предыдущий проект из обозревателя проектов.
Alt-F6	Активировать следующий проект из обозревателя проектов.
F11	Переключиться в редакторе между исходным <name>.cpp и соответствующим заголовочным файлом <name>.h.

1.11.6 Расширенные установки для компиляторов

В процессе сборки проекта сообщения компилятора отображаются в окне Messages закладки Build Log. Если вы хотите получить детальную информацию, отображение может быть расширено. Для этого щёлкните по 'Settings' →'Compiler and Debugger' и выберите 'Other Settings' в выпадающем поле.

Убедитесь, что выбран правильный компилятор. Установки 'Full command line' в поле Compiler Logging выводят полную информацию в Build Log. Дополнительно этот вывод можно записывать в HTML файл.

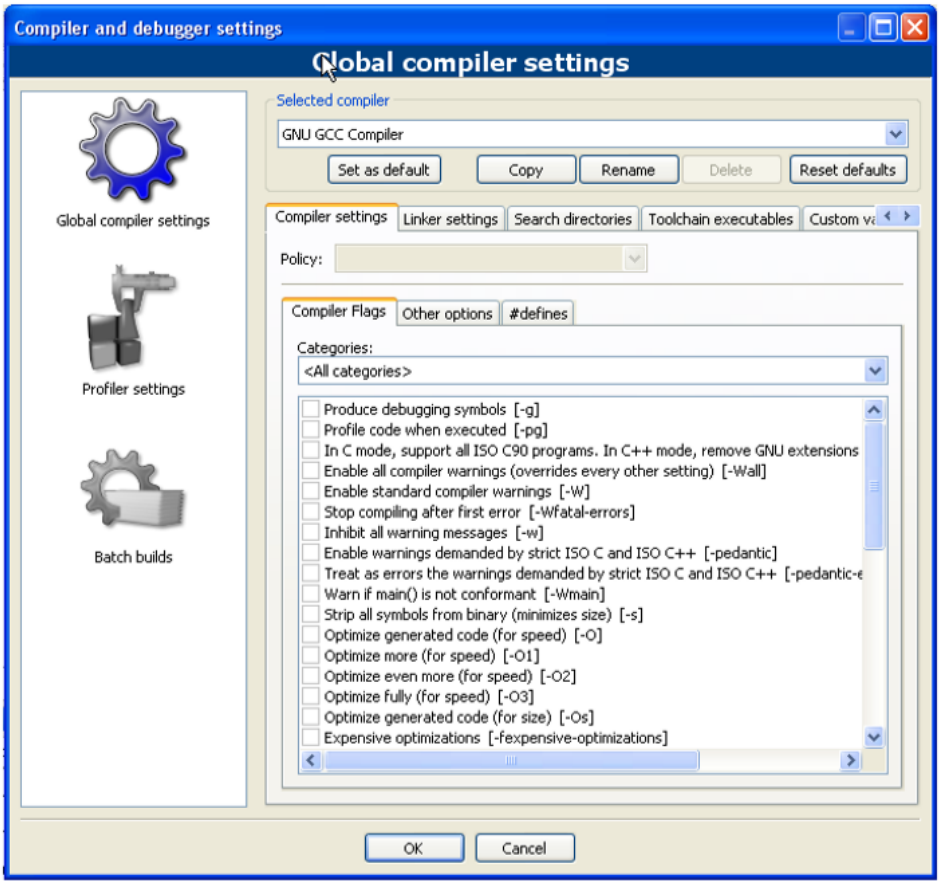


Рис. 1.13. Установка детальной информации

Для этого выберите ’Save build log to HTML file when finished’. Более того, CodeBlocks предлагает индикатор выполнения в окне Build Log, который может быть активирован установкой ’Display build progress bar’.

1.11.7 Масштабирование в редакторе

CodeBlocks предлагает весьма удобный редактор. Этот редактор позволяет вам изменять размер, в котором отображён открытый текст. Если вы используете мышку с колёсиком, вам нужно только нажать на клавишу **Ctrl** и повернуть колёсико, чтобы увеличить или уменьшить вид текста.

> Примечание:
>
> Горячими клавишами **Ctrl-Numepad-**/ или с помощью меню ’Edit’→’Special commands’ →’Zoom’ →’Reset’ оригинальный размер шрифта активного файла в редакторе будет перезаписан.

1.11.8 Режим оборачивания

При редактировании текстовых файлов *.txt в CodeBlocks может быт полезным «оборачивание» текста, означающее, что длинные строки будут отображаться на нескольких строках на экране, так чтобы они могли правильно редактироваться. Функция ’Word wrap’ может активироваться с помощью ’Settings’ →’Editor’ →’Other Options’ или установкой флажка ’Word wrap’. Клавиши **Home** и **End** перемещают курсор в начало и конец строки соответственно. Когда устанавливается ’Settings’ →’Editor’ →’Other Options’ и ’Home key always move to caret to first column’, курсор будет перемещаться в начало и конец текущей строки при нажатии на клавиши **Home** или **End**. Если

необходимо позиционировать курсор в начало первой строки текущего параграфа, используйте комбинацию горячих клавиш **Alt-Home**. Аналогично для позиционирования в конец последней строки текущего параграфа используйте **Alt-End**.

1.11.9 Выбор режимов в редакторе

CodeBlocks поддерживает разные режимы для выделения или вставки строк.

1. Левой клавишей мышки текст в активном редакторе может быть выделен, а затем клавишу можно отпустить. С помощью колёсика мышки пользователь может перемещаться к нужному месту. Если средняя клавиша мышки нажата, тогда ранее выделенный текст будет вставлен. Эта функция доступна в файле и может быть показана в буфере обмена в файле.

   Примечание: проверьте, не нужно ли использовать Copy после выделения текста.

2. Нажатие клавиши **ALT** активирует так называемый режим блочного выделения, а прямоугольное выделение реализовано левой клавишей мышки. Если клавиша **Alt** отпущена, это выделение можно скопировать и вставить. Эта возможность полезна, когда вы хотите выделить какую-нибудь колонку массива, скопировать её и вставить содержимое.
3. В меню 'Settings' →'Editor' →'Margins and Caret' может активироваться так называемый режим 'Virtual Spaces'. Эта опция делает возможным так, чтобы выделение в режиме выделения блока могло начинаться или завершаться на пустой строке.
4. В меню 'Settings' →'Editor' →'Margins and Caret'может быть активировано 'Multiple Selection'. Удерживая клавишу **Ctrl**, пользователь может выделять разные строки в активном редакторе с помощью левой клавиши мышки. Выделения будут добавляться в буфер обмена с помощью горячих клавиш **Ctrl-C** или **Ctrl-X**. **Ctrl-V** вставит содержимое в текущую позицию курсора. Дополнительно опция, названная 'Enable typing (and deleting)', может быть активирована для множественных выделений. Эта функция полезна, если вы хотите добавить директиву препроцессора подобную #ifdef на разные строки исходника, или если вы хотите переписать или заместить текст в нескольких местах.

Примечание:

Многие оконные менеджеры Linux используют **ALT-LeftClickDrag**, чтобы переместить окно, так что вам придётся отключить поведение этого оконного менеджера до того, как будет работать блочное выделение.

1.11.10 Свёртывание кода

CodeBlocks поддерживает так называемое свёртывание кода. То есть, вы можете свернуть функции в редакторе CodeBlocks. Точка сворачивания помечена символом минус в левой части окна редактора. На полях начало и конец от точки сворачивания видны как вертикальная линия. Если вы щёлкните по символу минус левой клавишей мышки, фрагмент кода будет свёрнут или развёрнут. С помощью меню 'Edit' →'Folding' вы можете выбрать сворачивание. В редакторе вы увидите свёрнутый код как длинную горизонтальную линию.

> Примечание:
>
> Стиль сворачивания и ограничение глубины могут конфигурироваться в меню ’Settings’ →’Editor’ →’Folding’.

CodeBlocks поддерживает возможность свёртывания также для препроцессорных директив. Чтобы разрешить эту функцию, установите флажок ’Fold preprocessor commands’ в меню ’Settings’ →’Editor’ на странице folding (выбор страниц диалога слева).

Другая возможность в том, чтобы задать определённые пользователем точки сворачивания. Начало точки сворачивания вводится как комментарий с открывающейся фигурной скобкой, а конец отмечается комментарием с закрывающейся скобкой.

```
//{ code with user defined folding //}
```

1.11.11 Автоматическое завершение

Если вы открываете проект в CodeBlocks, анализируются ’Search directories’ вашего компилятора и проекта, исходный и заголовочный файлы проекта. Вдобавок проверяются ключевые слова лексического анализатора. Полученная информация используется для автоматического завершения в CodeBlocks. Пожалуйста, проверьте настройки редактора, разрешена ли работа этой функции. Автозавершение доступно с помощью горячих клавиш **Ctrl-Space**. С помощью меню ’Settings’ →’Editor’ →’Syntax highlighting’ вы можете добавить определённые пользователем горячие клавиши в ваш лексический анализатор.

1.11.12 Поиск повреждённых файлов

Если файл удалён с диска, но ещё включён в список файлов проекта <project>.cbp, тогда этот ’broken file’ будет показан с символом повреждённого файла в обозревателя проекта. Вам нужно воспользоваться в меню ’Remove file from project, удалить файл из проекта’ вместо прямого удаления файлов.

В больших проектах с множеством поддиректорий поиск повреждённых файлов может занять много времени. CodeBlocks предлагает плагин ThreadSearch (см. раздел 2.6 на стр. 31), как простое решение этой проблемы. Если вы введёте выражение поиска в ThreadSearch и выберете опцию ’Project files’ или ’Workspace files’, тогда ThreadSearch будет анализировать все файлы, которые включены в проект или рабочее пространство. Если повреждённый файл обнаруживается, ThreadSerch выдаст ошибку с указанием пропущенного файла.

1.11.13 Включённые библиотеки

В опциях сборки проекта вы можете добавить используемые библиотеки с помощью кнопки **Add** в разделе ’Link libraries’ закладки ’Linker Settings’. Если так сделать, вы можете либо использовать абсолютный путь к библиотеке, либо просто задать имя без префикса lib и расширения файла.

Пример

Для вызова библиотеки <path>\libs\lib<name>.a, только впишите <name>. Компоновщик корректно включит библиотеки с соответствующими путями поиска

> Примечание:
>
> Другой способ включить библиотеки задокументирован в разделе 2.10 на стр. 40.

1.11.14 Порядок подключения объектных файлов

При компиляции создаётся объектный файл name.o из исходника name.c/cpp. Компоновщик затем связывает отдельные объектные файлы в приложение name.exe или для встроенной системы name.elf. В некоторых случаях может оказаться желательным предопределить порядок компоновки объектных файлов. В CodeBlocks это можно осуществить назначением приоритетов. В контекстном меню 'Properties' вы можете определить приоритеты файлов на закладке Build. Низкий приоритет означает, что файл будет связан раньше.

1.11.15 Автосохранение

CodeBlocks предлагает автоматическое сохранение проектов и исходных файлов или создание резервных копий. Эта функция может быть активирована в меню 'Settings' →'Environment' →'Autosave'. При этом следует задать 'Save to .save file' в качестве метода создания резервной копии.

1.11.16 Установки для расширений файлов

В CodeBlocks вы можете выбрать между несколькими способами обработки расширений файлов. Диалог настроек может быть открыт с помощью 'Settings' →'Files extension handling'. Вы можете либо использовать привязки приложений Windows для каждого из расширений файлов (открывать его с ассоциированным приложением), либо изменить эти установки для каждого из расширений так, чтобы либо запускалась определённая пользователем программа (запуск внешней программы), либо файл открывался в редакторе CodeBlocks (открыть его в редакторе Code::Blocks).

> Примечание:
>
> Если определённая пользователем программа связана с некоторым расширением файла, установка 'Disable Code::Blocks while the external program is running' будет деактивирована, поскольку иначе CodeBlocks будет закрываться, когда открывается файл с этим расширением.

1.12 CodeBlocks в командной строке

IDE CodeBlocks может выполняться из командной строки без графического интерфейса. В этом случае есть несколько доступных переключателей для управления процессом сборки проекта. Поскольку CodeBlocks поддерживает скрипты, создание исполняемого файла может интегрироваться в ваш рабочий процесс.

```
codeblocks.exe /na /nd --no-splash-screen --built <name>.cbp --
target='Release'
```

<filename> Задаёт имя проекта *.cbp или рабочего пространства *.workspace. Например, <filename> может быть project.cbp. Поместите аргумент в конце командной строки, но перед перенаправлением вывода, если таковое имеется.

--file=<filename>[:line]
Открывает файл в Code::Blocks и дополнительно выполняется переход на заданную строку.

/h, --help Показывает подсказку относительно аргументов командной строки.

/na, --no-check-associations
Не выполняется проверка ассоциации файла (только Windows).

/nd, --no-dde Не запускается DDE сервер (только Windows).

/ni, --no-ipc Не запускается IPC сервер (Linux и Mac только).

/ns, --no-splash-screen
Скрывает заставку начального экрана при загрузке приложения.

/d, --debug-log
Отображает сообщения отладки приложения.

--prefix=<str>
Устанавливает префикс директории общих данных.

/p, --personality=<str>, --profile=<str>
Устанавливает используемую персонализацию. Вы можете использовать запрос в качестве параметра для списка всех доступных персонализаций.

--rebuild Очищает и собирает проект или рабочее пространство.

--build Собирает проект или рабочее пространство.

--target=<str> Задаёт цель для пакетной сборки. Например **--target='Release'**.

--no-batch-window-close
Сохраняет видимым окно сообщений после завершения пакетной сборки.

--batch-build-notify
Показывает сообщение после завершения пакетной сборки.

--safe-mode Все плагины запрещаются при запуске.

> <build log file>
Помещается в самый конец командной строки, может использоваться для перенаправления стандартного вывода в файл журнала. Это не опция Codeblock как таковая, но только стандартное DOS/*nix shell перенаправление вывода.

1.13 Горячие клавиши

Даже если IDE, такое как в CodeBlocks, в основном поддерживается мышкой, горячие клавиши, тем не менее, остаются полезным инструментом для ускорения и упрощения процесса работы. В таблицах ниже мы собрали некоторые доступные горячие клавиши.

1.13.1 Editor

Function	Shortcut Key
Undo last action	Ctrl-Z
Redo last action	Ctrl-Shift-Z
Swap header / source	F11
Comment highlighted code	Ctrl-Shift-C
Uncomment highlighted code	Ctrl-Shift-X
Auto-complete / Abbreviations	Ctrl-Space/Ctrl-J
Toggle bookmark	Ctrl-B
Goto previous bookmark	Alt-PgUp
Goto next bookmark	Alt-PgDown

Это список горячих клавиш, поддерживаемых редактором CodeBlocks. Эти горячие клавиши не могут переназначаться.

Create or delete a bookmark	Ctrl-F2
Go to next bookmark	F2
Select to next bookmark	Alt-F2
Find selection.	Ctrl-F3
Find selection backwards.	Ctrl-Shift-F3
Find matching preprocessor conditional, skipping nested ones.	Ctrl-K

1.13.2 Files

Function	Shortcut Key
New file or project	Ctrl-N
Open existing file or project	Ctrl-O
Save current file	Ctrl-S
Save all files	Ctrl-Shift-S
Close current file	Ctrl-F4/Ctrl-W
Close all files	Ctrl-Shift-F4/Ctrl-Shift-W

1.13.3 View

Function	Shortcut Key
Show / hide Messages pane	F2
Show / hide Management pane	Shift-F2
Activate prior (in Project tree)	Alt-F5
Activate next (in Project tree)	Alt-F6

1.13.4 Search

Function	Shortcut Key
Find	Ctrl-F
Find next	F3
Find previous	Shift-F3
Find in files	Crtl-Shift-F
Replace	Ctrl-R
Replace in files	Ctrl-Shift-R
Goto line	Ctrl-G
Goto next changed line	Ctrl-F3
Goto previous changed line	Ctrl-Shift-F3
Goto file	Alt-G
Goto function	Ctrl-Alt-G
Goto previous function	Ctrl-PgUp
Goto next function	Ctrl-PgDn
Goto declaration	Ctrl-Shift-.
Goto implementation	Ctrl-.
Open include file	Ctrl-Alt-.

1.13.5 Build

Function	Shortcut Key
Build	Ctrl-F9
Compile current file	Ctrl-Shift-F9
Run	Ctrl-F10
Build and Run	F9
Rebuild	Ctrl-F11

2 Плагины

2.1 Astyle

Artistic Style (художественный стиль) – это «пробник» исходного кода, программа форматирования и украшения текста для C, C++, C# языков программирования. Он может использоваться для выбора разных стилей и правил кодирования в CodeBlocks.

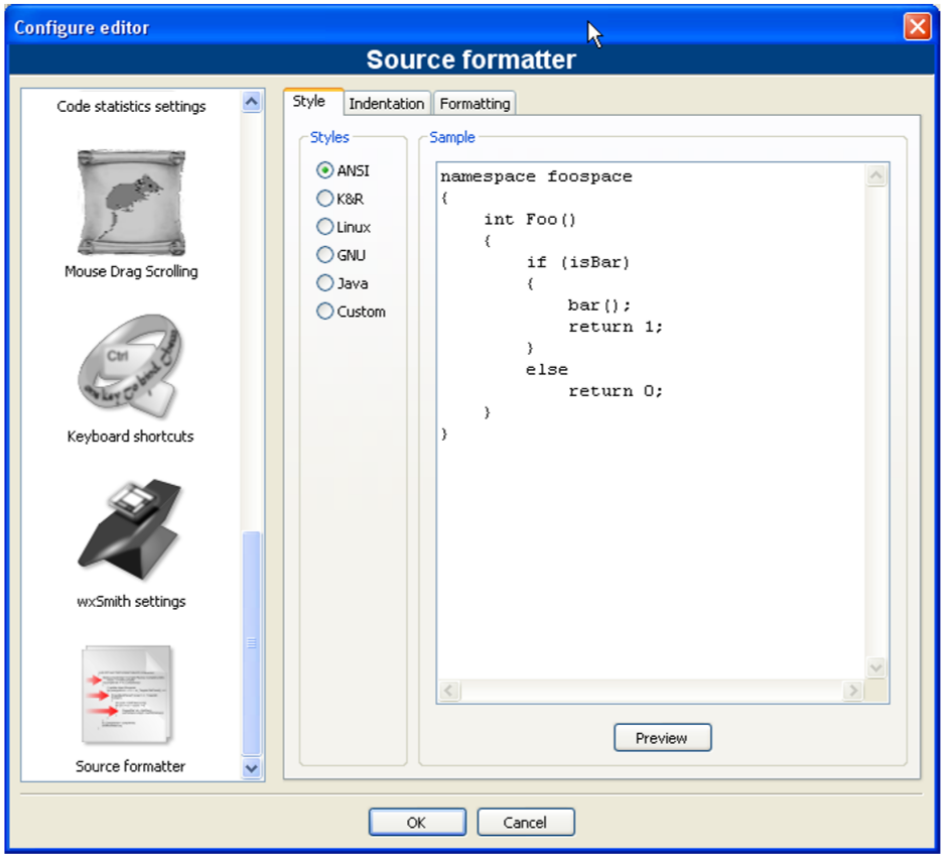


Рис. 2.1. Форматирование вашего исходного кода

Когда выравнивается исходный код, мы, как программисты, склонны использовать и пробелы, и табуляцию для создания желаемых отступов. Более того, некоторые редакторы по умолчанию вставляют пробелы вместо табуляции при нажатии на клавишу табуляции, а другие редакторы имеют возможность украшать строки, автоматически задавая пробелы перед кодом строки, возможность вставки пробелов в код, который ранее использовал только табуляцию для отступов.

Поскольку количество пробелов, показанное на экране для каждой табуляции в тексте исходного кода, меняется от редактора к редактору, одна из стандартных проблем программистов – оформление текста при переходе от одного редактора к другому. Проблема в том, что код, содержащий и пробелы, и табуляции, который до этого был с прекрасными отступами, внезапно превращается в беспорядочное нагромождение выражений при переходе к другому редактору. Даже если вы как программист тщательно используете ТОЛЬКО пробелы или табуляцию, просмотр исходного кода других людей всё ещё остаётся проблематичным.

Адресуясь к этим проблемам, был создан Artistic Style – фильтр, написанный на C++, который автоматически переделывает отступы и переформатирует C / C++ / C# исходные файлы.

> Примечание:
>
> При копировании кода, например, из Интернета или руководства этот код будет автоматически адаптирован к правилам кодирования в CodeBlocks.

2.2 CodeSnippets

CodeSnippets (фрагменты кода) – этот плагин делает возможным структурировать текстовые модули и соединять в файлы согласно с категориями в дереве вида. Модули используются для хранения часто используемых файлов, сборки их в текстовые модули и централизованного управления. Представьте себе следующую ситуацию: какое-то количество часто используемых исходных файлов хранятся в разных директориях файловой системы. Окно CodeSnippets предоставляет возможность создавать категории, а категории соединять с требуемыми файлами. С помощью этого вы можете управлять доступом к файлам независимо от того, где они хранятся в файловой системе, и вы можете быстро перемещаться между этими файлами без необходимости искать их по всей системе.

> Примечание:
>
> Вы можете использовать переменные CodeBlocks или переменные окружения в компоновке файлов, например, $(VARNAME)/name.pdf для параметризации связи в обозревателе CodeSnippets.

Список текстовых модулей может храниться в окне CodeSnippets, если щёлкнуть правой клавишей мышки и выбрать ’Save Index’ из контекстного меню. Файл codesnippets.xml, который будет создан этим процессом, можно позже найти в поддиректории codeblocks вашей директории Documents and Settings\Application data. В Linux эта информация хранится в поддиректории .codeblocks вашей директории HOME. Файлы конфигурации CodeBlocks будут загружены при следующем запуске программы. Если вы хотите сохранить содержимое CodeSnippets в другом месте, выберите пункт ’Save Index As’. Чтобы загрузить этот файл, выберите ’Load Index File’ при следующем запуске CodeBlocks или включите директорию в контекстном меню ’Settings’ под ’Snippet Folder’. Настройки сохранятся в соответствующем файле codesnippets.ini в ваших данных приложения.

Для включения подкатегории используйте меню ’Add SubCategory’. Категория может содержать Snippets (текстовые модули) или File Links (ссылки на файлы). Текстовый модуль создаётся с помощью команды ’Add Snippet’ в контекстном меню. Содержимое интегрируется в текстовый модуль как ’New snippet’ при выделении текста в редакторе CodeBlocks и перетаскивании его (drag and drop) на модуль, после чего открывается диалог свойств. Дважды щёлкните по вновь включённому объекту или выберите ’Edit Text’, чем откроете редактор с содержимым.

Вывод текстового модуля поддерживается в CodeBlocks через команду контекстного меню ’Apply’ или перетаскиванием в редактор. В Windows содержимое Snippet может также перетаскиваться в другие приложения. В обозревателе CodeSnippets Browser вы можете копировать выбранные объекты с последующим перетаскиванием в разные категории.

Помимо этого текстовые модули могут параметризироваться переменными <name>, к которым есть доступ через $(name) (см. рис. 2.2 на стр. 27). Значения переменных можно отыскать в поле ввода, если текстовый модуль вызывается командой контекстного меню 'Apply'.

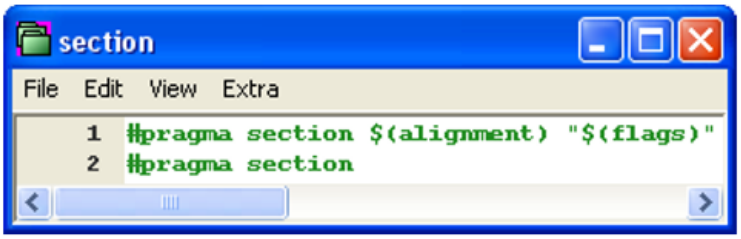


Рис. 2.2. Редактирование текстового модуля

Кроме текстовых модулей могут создаваться связи между файлами. Если после создания текстового модуля вы щёлкните по команде контекстного меню 'Properties', тогда вы сможете выбрать цель ссылки щелчком по кнопке **Link target**. Эта процедура будет автоматически конвертировать текстовый модуль в ссылку на файл. В CodeSnippets все текстовые модули будут маркироваться символом T, ссылка на файл символом F, а url символом U. Если вы хотите открыть выбранный файл (ссылку) в обозревателе codesnippets, достаточно выбрать в контекстном меню 'Open File' или удержать клавишу **Alt** и дважды щёлкнуть по файлу.

> Примечание:
>
> Вы может добавить и url (то есть, http://www.codeblocks.org) в текстовые модули. url можно открывать с помощью контекстного меню 'Open Url' или перетаскиванием в ваш любимый web-обозреватель.

С этими настройками, если открыть ссылку на pdf файл из обозревателя codesnippets, автоматически запустится обозреватель pdf. Этот метод делает возможным использование доступа к файлам, которые распространяются по всей сети, таким как данные CAD, планы размещения, документация и т.д., общими приложениями просто с помощью ссылки. Содержимое codesnippets хранится в файле codesnippets.xml, конфигурация хранится в файле codesnippets.ini в вашей директории данных приложений. Этот ini файл, например, содержит путь файла codesnippets.xml.

CodeBlocks поддерживает использование разных профилей. Эти профили называются персонализациями. Запуская CodeBlocks из командной строки с опцией *--personality=<profile>,* вы создадите новый или используете существующий профиль. Эти установки не будут храниться в файле default.conf, но в <personality>.conf в вашей директории данных приложений. Плагин Codesnippets будет хранить свои настройки в файле <personality>.codesnippets.ini. Теперь, если вы загрузите новое содержание <name.xml> в установки Codesnippets с помощью 'Load Index File', это содержимое будет храниться в соответствующем ini файле. Преимущество этого метода состоит в том, что при разных профилях разные конфигурации текстовых модулей и ссылок будут работать.

Плагин предлагает дополнительную функцию поиска для навигации между категориями и Snippets. Граница поиска Snippets – это категории или текстовые модули (Snippets), а категории могут подстраиваться. При вводе требуемого выражения для поиска соответствующий ввод

автоматически выбирается в обозревателе. Рисунок 2.3 на странице 28 показывает типичное отображение в окне CodeSnippets.

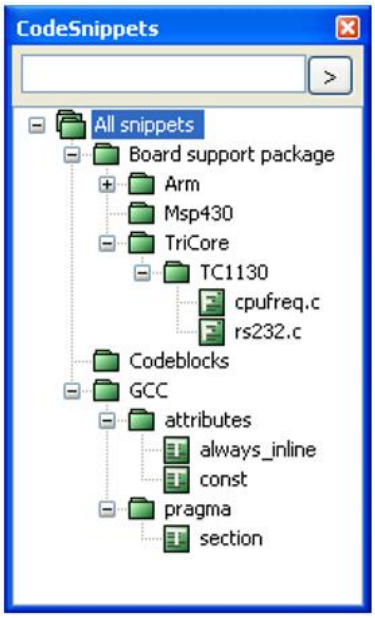


Рис. 2.3. Обозреватель CodeSnippets

Примечание:

При использовании многотомных текстовых модулей содержание этих модулей будет сохраняться в файле через 'Convert to File Link' в плане уменьшения потребности в используемой в системе памяти. Если вы удалите codesnippet или ссылку на файл, он будет перемещён в категорию .trash; если вы удержите клавишу **Shift**, объект будет удалён.

2.3 Incremental Search

Для эффективного поиска в открытых файлах CodeBlocks предлагает так называемый инкрементальный поиск, Incremental Search. Этот метод поиска инициируется для открытого файла с помощью меню 'Search' →'Incremental Search' или горячими клавишами **Ctrl-I**. Акцент делается на автоматической установке маски соответствующей инструментальной панели. Как только вы начнёте ввод в окно поиска, фоновый поиск с маской будет подстраиваться к соответствию с обнаруженными объектами. Если совпадение найдётся в активном редакторе, соответствующая позиция будет отмечена в тексте цветом. По умолчанию текущее совпадение будет подсвечиваться зелёным. Эту настройку можно изменить в 'Settings' →' Editor' →' Incremental Search' (см. ?? на стр. ??). Нажав клавишу **Return**, вы стимулируете поиск, чтобы перейти к следующему обнаружению строки в файле. Клавишами **Shift-Return** может быть выделено предыдущее обнаружение. Эта функциональность не поддерживается Scintilla, если инкрементальный поиск использует регулярные выражения.

```
m_pToolbar->EnableTool(X

if (m_pControl != 0)
{
    m_SearchText=m_pText
    m_pToolbar->EnableTo
    m_pToolbar->EnableTo
    m_NewPos=m_pControl-
    m_OldPos=m_NewPos;
}
else
{
    m_pToolbar->EnableTo
    m_pToolbar->EnableTo
}
```

Если строка поиска не может быть найдена в активном файле, этот факт подсвечивается фоном маски поиска, который станет красным.

ESC Покинуть модуль Incremental Search.

ALT-DELETE Очистить поле ввода инкрементального поиска.

Иконки инструментальной панели Incremental Search имеют следующее назначение:

Удаление текста в маске поиска инструментальной панели Incremental Search.

Навигация между обнаруженными совпадениями со строкой поиска.

Щелчок по этой кнопке приведёт к окрашиванию всех найденных в редакторе строк, а не только первоначальное обнаружение.

Активизация этой опции ограничивает поиск в текстовом отрывке, отмеченном в редакторе.

Эта опция означает, что выполнится чувствительный к регистру поиск.

Регулярное выражение может быть использовано в поле ввода инкрементального поиска.

Примечание: Стандартные установки этой инструментальной панели могут быть сконфигурированы в ’Settings’ →’Editor’ →’Incremental Search’.

2.4 ToDo List

В сложных программных проектах, где задействованы разные пользователи, часто есть потребность в разных задачах, которые будут выполнять разные пользователи. Для этой цели CodeBlocks предлагает Todo List (список намеченных дел). Этот список может быть открыт с помощью 'View' →'To-Do list', а содержание задач для выполнения вместе с их приоритетами вписываются ответственными за это пользователями. Список может фильтроваться по задачам, пользователям и/или исходным файлам. Сортировка в колонках может быть выполнена щелчком по заголовку соответствующей колонки.

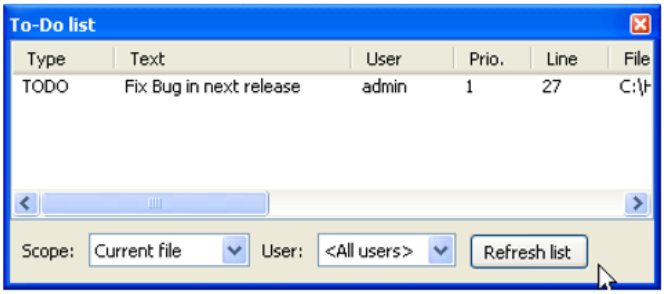


Рис. 2.4. Отображение ToDo List

> Примечание:
>
> To-Do list может быть прикреплён в консоли сообщений. Выберите опцию 'Include the To-Do list in the message pane' с помощью меню 'Settings' →'Environment'.

Если исходники открыты в CodeBlocks, Todo может быть добавлено к списку через контекстное меню командой 'Add To-Do item'. Комментарий будет добавлен в выбранную строку исходного кода.

```
// TODO (user#1#): add new dialog for next release
```

При добавлении To-Do открывается диалоговое окно, где могут быть сделаны следующие установки (см. рис. 2.5 на стр. 31).

User Имя пользователя в операционной системе. Задания для других пользователей также могут быть созданы здесь. Чтобы это сделать, соответствующее имя пользователя должно быть создано с помощью **Add new**. Присваивание в Todo затем выполняется через выбор элементов списка User.

> Примечание:
>
> Заметьте, что Users не имеют ничего общего с Personalities, которые используются в CodeBlocks.

Type По умолчанию type задаётся в Todo.

Priority Значительность задач может быть определена приоритетами (1 - 9) в CodeBlocks.

Position Этим задаётся, будет ли комментарий включён до, после или сразу за курсором.

Comment Style Выбор форматов для комментариев (например, doxygen).

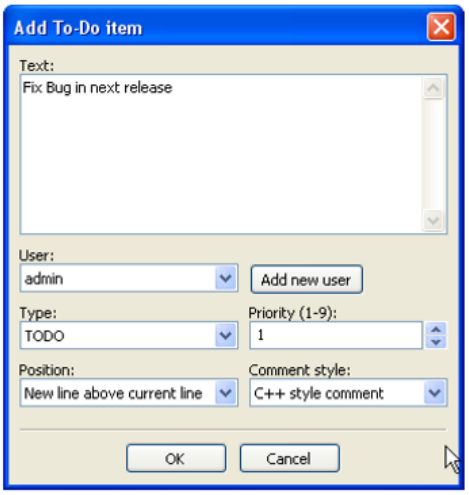


Рис. 2.5. Диалог для добавления ToDo

2.5 Source Code Exporter

Часто появляется необходимость в передаче исходного кода в другие приложения или по e-mail. Если текст просто копировать, форматирование будет потеряно, в результате текст станет плохо читаем. Функция экспорта CodeBlocks используется в качестве лекарства в подобной ситуации. Требуемый формат для экспорта файла может быть выбран в 'File' →'Export'. Программа затем примет имя файла и целевую директорию из открытого исходного файла и использует для сохранения экспортируемого файла. Подходящее расширение файла в каждом случае будет определено форматом экспорта. Доступны следующие форматы.

html Текстовый формат, который может отображаться в web-обозревателе или текстовом процессоре.

rtf Rich Text формат, текстовый формат, который может быть открыт в текстовом процессоре, таком как Word или OpenOffice.

odt Open Document Text формат, стандартный формат, который был создан Sun и O'Reilly. Этот формат может читаться Word, OpenOffice и другими текстовыми процессорами.

pdf Portable Document Format может открываться такими приложениями как Acrobat Reader.

2.6 Thread Search

Через меню 'Search'→'Thread Search' предназначенный плагин может быть показан или скрыт как закладка в Messages Console. В CodeBlocks предварительный просмотр может быть выведен для обнаружения символьной строки в файле, рабочем пространстве или директории. Чтобы это сделать, список результатов поиска будет отображаться справа в ThreadSearch Console (консоль поточного поиска). Щелчок по записи в списке вызовет появление предварительного просмотра слева. Двойной щелчок в списке откроет выбранный файл в редакторе CodeBlocks.

> Примечание:
>
> Граница расширений файлов, включаемых в поиск, предустановлена и, возможно, должна быть подкорректирована.

2.6.1 Возможности

Плагин ThreadSearch предоставляет следующие возможности:

- Многопоточный 'Search in files'.
- Внутренний редактор (только для чтения) для предварительного просмотра результатов.
- Открывание файла в редакторах notebook .
- Контекстное меню 'Find occurrences', чтобы начать поиск в файлах со слова под курсором.

Рис. 2.6. Панель Thread Search

2.6.2 Использование

1. Сконфигурируйте ваши предпочтения при поиске (см. рис. 2.7 на стр. 33)

   Как только плагин установлен, есть 4 варианта запуска поиска:

   a) Введите/выделите слово в окне поиска и нажмите **Enter** или щёлкните по **Search** на панели Thread Search блокнота Messages.
   b) Введите/выделите слово инструментальной панели окна поиска и нажмите **Enter** или щёлкните по кнопке **Search**.
   c) Щёлкните правой клавишей мышки по любому «слову» в активном редакторе и щёлкните по 'Find occurrences'.
   d) Щёлкните по Search/Thread поиску, чтобы найти текущее слово в активном редакторе.

   > Примечание:
   >
   > Пункты 1, 2 и 3 могут быть недоступны в текущей конфигурации.

2. Щёлкните ещё раз по кнопке **Search**, чтобы прервать текущий поиск.
3. Единичный щелчок по результату отобразит его в редакторе предпросмотра в нужном месте.
4. Двойной щелчок по результату откроет элемент или установит редактор в блокноте редактора в нужное место.

2.6.3 Конфигурация

Для доступа к панели конфигурации ThreadSearch щёлкните по (см. рис. 2.7 на стр. 33):

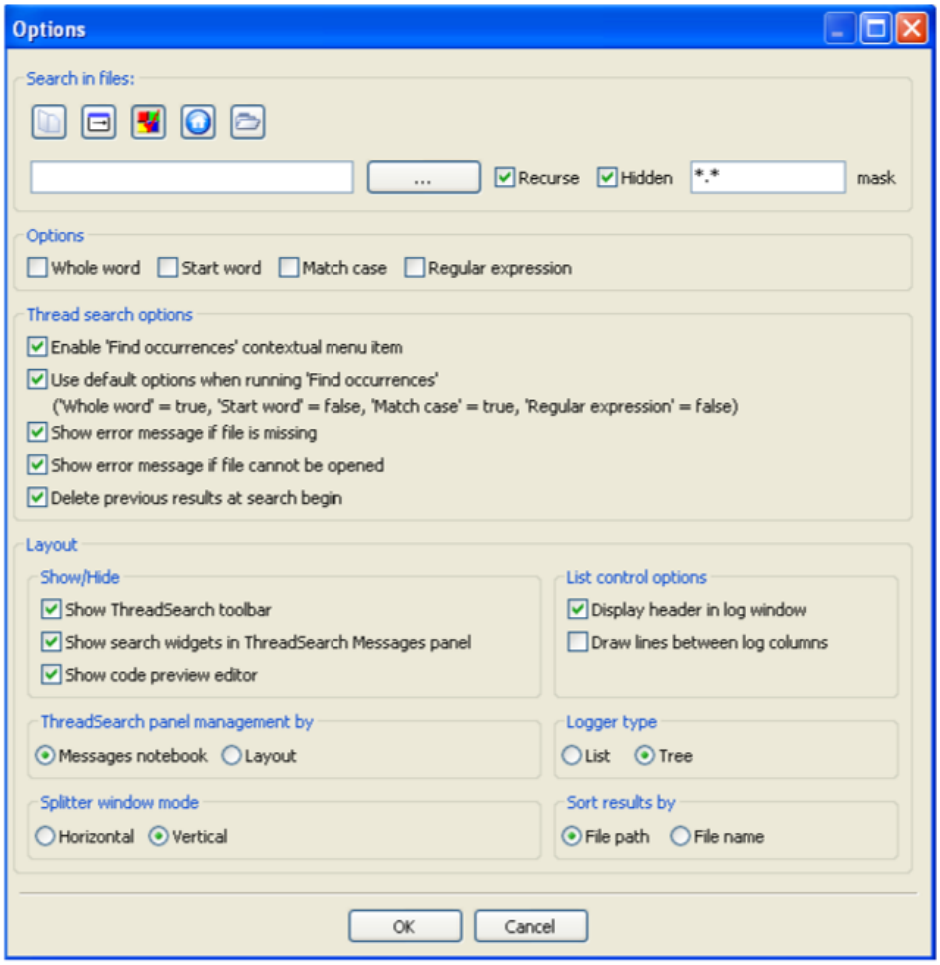


Рис. 2.7. Конфигурация Thread Search

1. Кнопке **Options** на панели Messages notebook Thread search.
2. Кнопке **Options** на Thread search инструментальной панели.
3. Settings/Environment разделу меню, а затем по Thread search в левой колонке.

Примечание: Пункты 1, 2 и 3 могут быть недоступны в текущей конфигурации.

Поиск частично зависит от набора файлов, которые будут анализироваться.

- Флажки проекта и рабочего пространства в переключателях (checkboxes) взаимно исключающие.
- Путь к директории может быть отредактирован или установлен кнопкой **Select**.
- Маска задаёт спецификации файлов, разделённые ';'. Например: *.cpp;*.c;*.h.

2.6.4 Опции

Whole word если отмечено, ищется совпадающая строка с выражением, если искомое выражение найдено без буквенно-цифрового +'_' перед или после.

Start word если отмечено, ищется совпадающая строка с выражением, если искомое

выражение найдено в начале слова, то есть, без буквенно-цифрового +'_' перед выражением поиска.

Match case если отмечено, поиск чувствителен к регистру.

Regular expression выражение поиска – регулярное выражение.

> Примечание:
>
> Если вы хотите искать регулярное выражение, наподобие n, вы должны установить опцию 'Use Advanced RegEx searches' через меню 'Settings' →'Editor' →'General Settings'.

2.6.5 Thread Search опции

Enable 'Find occurrences contextual menu item' если отмечено, *Find occurrences* из ввода 'Focused word' добавляется к контекстному меню.

Use default options when running 'Find occurrences' если отмечено, набор опций по умолчанию применяется к работе поиска с разделом 'Find occurrences' контекстного меню. Опции по умолчанию 'Whole word' и 'Match case' разрешены.

Delete previous results at search begin если Thread Search сконфигурировано с 'Tree View' тогда результаты поиска будут в иерархическом списке

- первый узел содержит слово поиска;
- выше перечислены файлы, которые содержат слово поиска;
- внутри списка номер строк и соответствующее содержание обнаруженного.

Если вы ищете разные слова, список станет сбивать с толку, поэтому результаты предыдущих поисков могут быть очищены перед поиском с этой опцией.

> Примечание:
>
> В списке обнаруженного один из объектов или все объекты могут удаляться с помощью контекстного меню 'Delete item' или 'Delete all items'.

2.6.6 Внешний вид

Display header in log window если отмечено, заголовок отображается в списке результатов.

> Примечание:
>
> Если не отмечено, колонки не будут изменяемого размера, но место экономится.

Draw lines between columns проводить линии между колонками в режиме списка.

Show ThreadSearch toolbar отображать инструментальную панель плагина Thread Search.

Show search widgets in ThreadSearch Messages panel если отмечено, отображается только управление списком результатов и редактор предпросмотра. Все остальные виджеты скрыты (экономия места).

Show code preview editor предпросмотр кода может быть скрыт либо этим флажком переключателя, либо двойным щелчком по разделителю окон в середине. Там же вновь можно восстановить видимость.

2.6.7 Панель Management

Вы можете выбрать разные режимы, как управлять окном ThreadSearch. Установкой 'Message Notebook' окно ThreadSearch будет прикреплённым к панели сообщений. Если вы отметили установку 'Layout', вам будет доступно открепить окно от панели сообщений и поместить его где-то ещё.

2.6.8 Регистратор типа

Вид результатов поиска может быть разным. Установка 'List' отображает все обнаружения в виде списка. Другой режим 'Tree' собирает всё обнаруженное в файле в виде узловой точки.

2.6.9 Режим разделения окон

Пользователь может конфигурировать горизонтальное или вертикальное разделение окон предпросмотра и окна вывода результатов поиска.

2.6.10 Результаты краткого поиска

Вид результатов поиска может быть отсортирован по пути или по имени файла.

2.7 FileManager и PowerShell плагин

Проводник (File Explorer), рисунок 2.8 на странице 36, включён в плагин FileManager, и может быть найден на закладке 'Files'. Композиция File Explorer показана на рисунке 2.8 на странице 36.

В верхней части вы найдёте поле для ввода пути. Щелчком по кнопке в конце этого поля выпадающее окно перечислит историю последних вводов, по которому можно перемещаться с помощью полосы прокрутки. Стрелка вверх на правой стороне поля перемещает вверх структуру директорий на одну директорию.

В поле 'Wildcard' вы можете ввести выражение фильтра для отображения файла. Оставив поле пустым или введя ∗, вы отобразите результаты всех файлов. Ввод ∗.c;∗.h, например, отобразит результат просмотра исходных Си и заголовочных файлов. Открыв выпадающее окно, вы вновь обнаружите историю всех последних вводов.

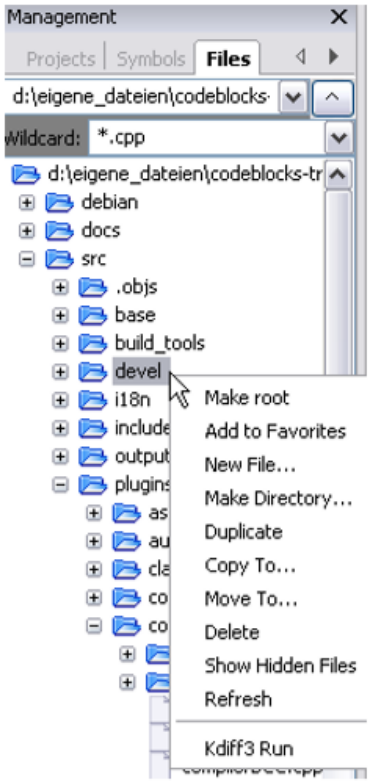


Рис. 2.8. Файловый менеджер

При нажатой клавише **Shift** щелчком мышки можно выделить группу файлов или директорий, нажатая клавиша **Ctrl** с последующими щелчками выделяет множество отдельных файлов или директорий.

Следующие операции можно начать с помощью контекстного меню, если одна или множество директорий выделены в File Explorer:

Make Root определяет текущую директорию в качестве корневой (root) директории.

Add to Favorites ставит маркер на директорию и сохраняет её к избранную. Эта функция позволяет вам быстро перемещаться между часто используемыми директориями, а также и на разных сетевых дисках.

New File создаёт новый файл в выбранной директории.

New Directory создаёт новую поддиректорию в выбранной директории.

Следующие операции можно начинать через контекстное меню, если один или множество файлов или директорий выделено в File Explorer:

Duplicate копирует файл/директорию и переименовывает их.

Copy To открывает диалог для ввода целевой директории, в которую будет копироваться файл/директория для хранения.

Move To перемещает выбранное в целевое место.

Delete удаляет выбранные файлы/директории.

Show Hidden Files активирует/деактивирует отображение скрытых системных файлов. Когда активировано, этот пункт меню отмечен.

Refresh обновляет отображение дерева директорий.

Следующие операции могут начинаться с помощью контекстного меню, если один или множество файлов выделены в File Explorer:

Open in CB Editor открывает выделенный файл в CodeBlocks редакторе.

Rename переименовывает выделенный файл.

Add to active project добавляет файл(ы) к активному проекту.

> Примечание:
>
> Файл/директория, выделенные в File Explorer, могут ассоциироваться в плагине PowerShell через переменную mpaths.

Определённые пользователем функции могут быть заданы командой меню 'Settings'→'Environment' →'PowerShell'. В маске PowerShell новая функция, которая может быть названа случайным образом, создаётся с помощью кнопки **New**. В поле 'ShellCommand Executable' запускается выполняемая программа, а в поле в нижней части окна могут передаваться дополнительные параметры в программу. Щелчком по функции в контекстном меню или меню PowerShell menu функция запускается, а затем обрабатывает выделенные функции/директории. Вывод перенаправляется в отдельное окно оболочки.

Например, элемент меню 'PowerShell' →'SVN' в контекстном меню создаётся для 'SVN'. $file в этом контексте означает файл, выделенный в File Explorer, $mpath выделенные файлы или директории (см. раздел 3.2 на стр. 54).

```
Add;$interpreter add $mpaths;;;
```

Эта и каждая последующая команда будут создавать подменю, в этом случае вызывая 'Extensions' →'SVN' →'Add'. Контекстное меню соответственно расширяется. Щелчок по команде в контекстном меню выполнит SVN команду и обработает выделенные файлы/директории.

TortoiseSVN – это широко распространённая SVN программа с интеграцией в проводник. Программа TortoiseProc.exe от TortoiseSVN может запускаться в командной строке и отображать диалог для получения данных, вводимых пользователем. Так что вы можете выполнять команды, которые доступны как контекстное меню в проводнике, как и в командной строке. Поэтому вы можете интегрировать это shell расширение в CodeBlocks. Например, команда

```
TortoiseProc.exe /command:diff /path:$file
```

выделит diff файл в CodeBlocks файловом менеджере с базой SVN. Смотрите рисунок 2.9 на странице 38, как интегрировать эту команду.

Примечание:

Для файлов под управлением SVN файловый менеджер покажет наложенные иконки, если они активированы через меню 'View' →'SVN Decorators'.

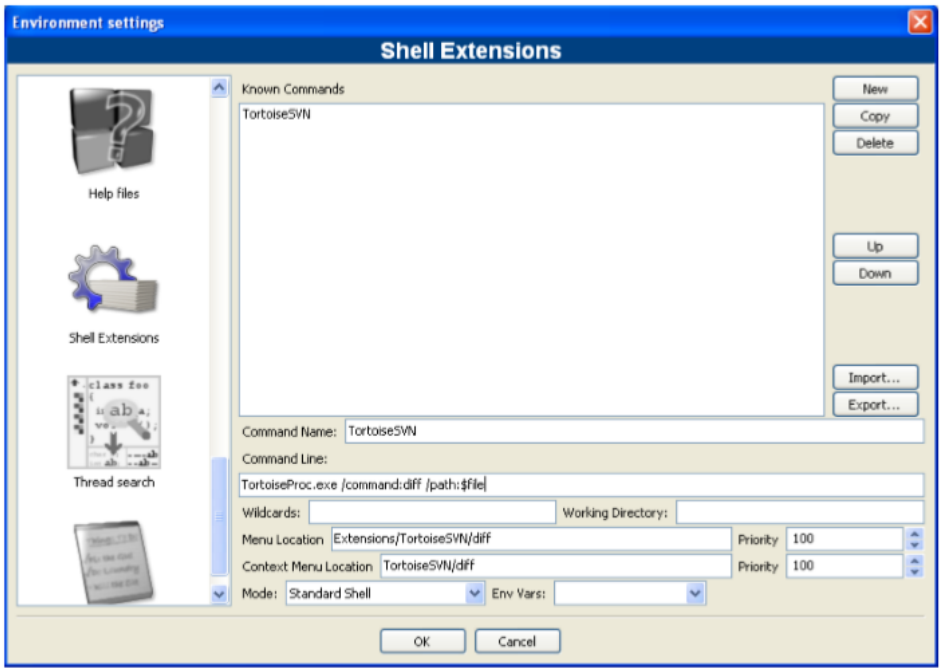


Рис. 2.9. Добавление расширения shell в контекстное меню

Пример

Вы можете использовать файловый менеджер для diff файлов или директорий. Выполните следующие шаги:

1. Добавьте имя через меню 'Settings' →'Environment' →'PowerShell'. Это появится как пункт в меню интерпретатора и контекстном меню.
2. Выделите абсолютный путь исполняемого Diff (kdiff3). Программа доступна с переменной $interpreter .
3. Добавьте параметры интерпретатора

```
Diff;$interpreter $mpaths;;;
```

Эта команда будет выполняться, используя выделенные файлы или директории как параметры. Выделение доступно через переменную $mpaths. Это лёгкий путь к файлам diff или директориям.

Примечание:

Плагин поддерживает использование CodeBlocks переменных в shell расширении.

$interpreter Вызывает этот исполняемый файл.

$fname Имя файла без расширения.

$fext Расширение выделенного файла.

$file Имя файла.

$relfile Имя файла без информации о пути.

$dir Имя выделенной директории.

$reldir Имя директории без информации о пути.

$path Абсолютный путь.

$relpath Относительный путь файла или директории.

$mpaths Список текущих выделенных файлов или директорий.

$inputstr{<msg>} Строка, которая введена в окно сообщения.

$parentdir Родительская директория (../).

> Примечание:
>
> Элементы shell расширения также доступны как контекстное меню в редакторе CodeBlocks.

2.8 Browse Tracker

Browse Tracker – это плагин, который помогает перемещаться между недавно открытыми файлами в CodeBlocks. Список недавно открытых файлов хранится в history (история). В меню 'View' →'Browse Tracker' →'Clear All' история очищается. С окном 'Browsed Tabs' вы можете перемещаться между пунктами недавно открытых файлов, воспользовавшись разделом меню 'View' →'Browse Tracker' →'Backward Ed/Forward Ed' или горячими клавишами **Alt-Left/Alt-Right**. Меню Browse Tracker также доступно через контекстное меню. Маркеры хранятся в файле внешнего вида <projectName>.bmarks.

Общая процедура при создании программы – это битва с множеством функций, которые реализованы в разных файлах. Плагин BrowseTracks поможет вам решить эту проблему, показав вам порядок, в котором файлы выбирались. Вы можете затем удобно перемещаться по вызовам функции.

Плагин позволяет даже просматривать маркеры в каждом файле в редакторе CodeBlocks. Положение курсора запоминается для каждого файла. Вы можете задать эти маркеры, используя пункты меню 'View' →' Browse Tracker' →' Set BrowseMarks' или выделяя строку с помощью левой клавиши мышки. Маркер с ... показан в левом поле. С помощью меню 'View' →'Browse Tracker' →'Prev Mark/Next Mark' или горячих клавиш **Alt-up/Alt-down** вы можете перемещаться по маркерам в файле. Если вы хотите перемещаться в файле между маркерами, отсортированными по номерам строк, тогда только выберите меню 'View' →'Browse Tracker' →'Sort BrowseMark'. С 'Clear BrowseMark' маркер в выделенной строке удаляется. Если маркер установлен на строке, удерживание нажатой левой клавиши мышки на ¼ секунды при нажатой клавише **Ctrl** удалит маркер с этой строки. С помощью меню 'Clear All BrowseMarks' или **Ctrl-left** щелчка по любой не маркированной строке можно сбросить маркеры в файле.

Установки плагина могут конфигурироваться с помощью меню 'Settings' →'Editor' →'Browse Tracker' .

Mark Style Просмотр отметок (Browse Marks) отображается по умолчанию как ... на полях. Установкой 'Book Marks' они будут отображаться подобно Bookmarks как синяя стрелка на полях. С «hide» отображение Browse Marks подавляется.

Toggle Browse Mark key Маркеры могут устанавливаться или удаляться либо щелчком левой клавиши мышки, либо щелчком при удержании клавиши **Crtl**.

Toggle Delay Длительность удержания левой клавиши мышки для входа в режим Browse Marker.

Clear All BrowseMarks При удержании клавиши **Ctrl** либо простым, либо двойным щелчком левой клавиши мышки.

Конфигурация плагина хранится в директории ваших Application Data в файле default.conf. Если вы используете функцию персонализации CodeBlocks, конфигурация читается из файла <personality>.conf.

2.9 SVN поддержка

Поддержка системы контроля версии SVN включена в CodeBlocks плагин TortoiseSVN. С помощью меню 'TortoiseSVN' →'Plugin settings' вы можете конфигурировать доступные svn команды на закладке 'Integration'.

Menu integration Добавляет запись TortoiseSVN с разными установками в меню.

Project manger Активирует команды TortoiseSVN в контекстном меню менеджера проекта.

Editor Активизирует команды TortoiseSVN в контекстном меню редактора.

В настройках плагина вы можете установить, какие команды svn доступны в меню или контекстном меню. Закладка integration поддерживает пункты 'Edit main menu' и 'Edit popup menu' для конфигурирования этих команд.

> Примечание:
>
> File Explorer в CodeBlocks использует разные наложенные иконки, чтобы показать состояние svn. Команды TortoiseSVN включены здесь в контекстное меню.

2.10 LibFinder

Если вы хотите использовать какие-то библиотеки в вашем приложении, вы должны сконфигурировать ваш проект для их использования. Подобный процесс конфигурации может оказаться утомительным и раздражающим, поскольку каждая из библиотек может использовать схему пользовательских опций. Другая проблема в том, что конфигурация зависит от платформы, результаты в unix и windows проектах несовместимы.

LibFinder имеет две основные функциональные особенности:

- Поиск библиотек установлен в вашей системе.

- Включение библиотеки в ваш проект только несколькими щелчками мышки делает проект независимым от платформы.

2.10.1 Поиск библиотек

Поиск библиотек доступен через меню 'Plugins' →'Library finder'. Его цель – обнаружить библиотеки в вашей системе и сохранить результаты в базе данных LibFinder (заметьте, что эти результаты не записываются в файлы проекта CodeBlocks). Поиск начинается с диалога, где вы можете предоставить набор директорий с установленными библиотеками. LibFinder будет сканировать их рекурсивно, так что, если вы не уверены, вы можете выбрать какие-то базовые директории. Вы можете даже добавить весь диск – в этом случае процесс поиска займёт больше времени, но может обнаружить больше библиотек (см. рис. 2.10 на стр. 41).

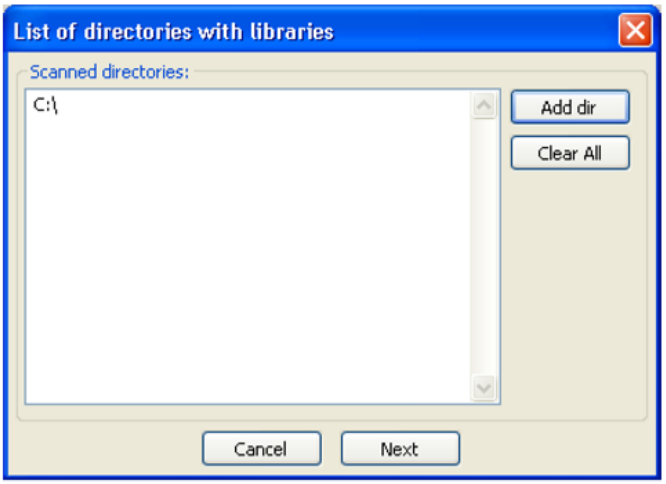


Рис. 2.10. Список директорий

Когда LibFinder сканирует директории в поисках библиотек, используются специальные правила обнаружения наличия библиотеки. Каждый набор правил локализован в xml файле. В настоящее время LibFinder может искать wxWidgets 2.6/2.8, CodeBlocks SDK и GLFW. Список будет в дальнейшем пополняться.

> Примечание:
>
> Чтобы получить более детальные данные, как добавить библиотеки, поддерживаемые в LibFinder, прочитайте src/plugins/contrib/lib finder/lib finder/readme.txt в исходниках CodeBlocks.

После завершения сканирования LibFinder покажет результаты (см. рис. 2.11 на стр. 42).

В списке вы отмечаете библиотеки, которые будут храниться в базе данных LibFinder. Заметьте, что каждая библиотека может иметь более, чем одну правильную конфигурацию, а настройки, добавленные ранее, чаще используются при сборке проектов.

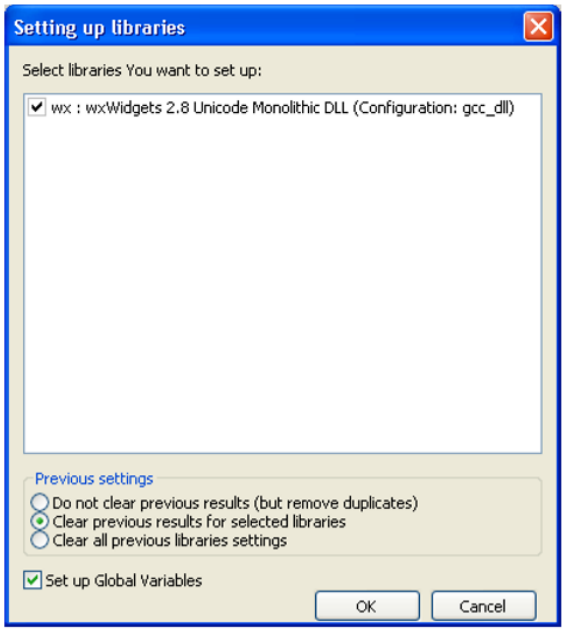


Рис. 2.11. Результаты поиска

Ниже список того, что вы можете выбрать для использования результатов предыдущего сканирования:

Do not clear previous results Эта опция работает подобно обновлению существующих результатов – добавляет новые и обновляет те, которые уже существуют. Эта опция не рекомендуется.

Second option (Clear previous results for selected libraries) будет удалять все результаты библиотек, которые выбраны ранее перед добавлением новых результатов. Это рекомендованная опция.

Clear all previous library settings когда вы выбираете эту опцию, база данных LibFinder будет очищена перед добавлением новых результатов. Это полезно, когда вы хотите очистить базу данных LibFinder от каких-то неверных записей.

Другая опция этого диалога в 'Set up Global Variables'. Когда вы задаёте эту опцию, LibFinder постарается автоматически сконфигурировать Global Variables, которые также используются в помощи при обработке библиотек.

Если у вас в системе установлено pkg-config (устанавливается автоматически на большинстве версий Linux), LibFinder будет также поддерживать библиотеки этого инструмента. Нет необходимости выполнять сканирование для них – они автоматически загружаются, когда запускается CodeBlocks.

2.10.2 Включение библиотек в проекты

LibFinder добавляет дополнительную закладку в Project Properties 'Libraries' – эта закладка показывает библиотеки, используемые в проекте, и библиотеки, которые известны LibFinder. Чтобы добавить библиотеку в ваш проект, выберите её на правой панели и щёлкните кнопку **>** (см. рис. 2.12 на стр. 43).

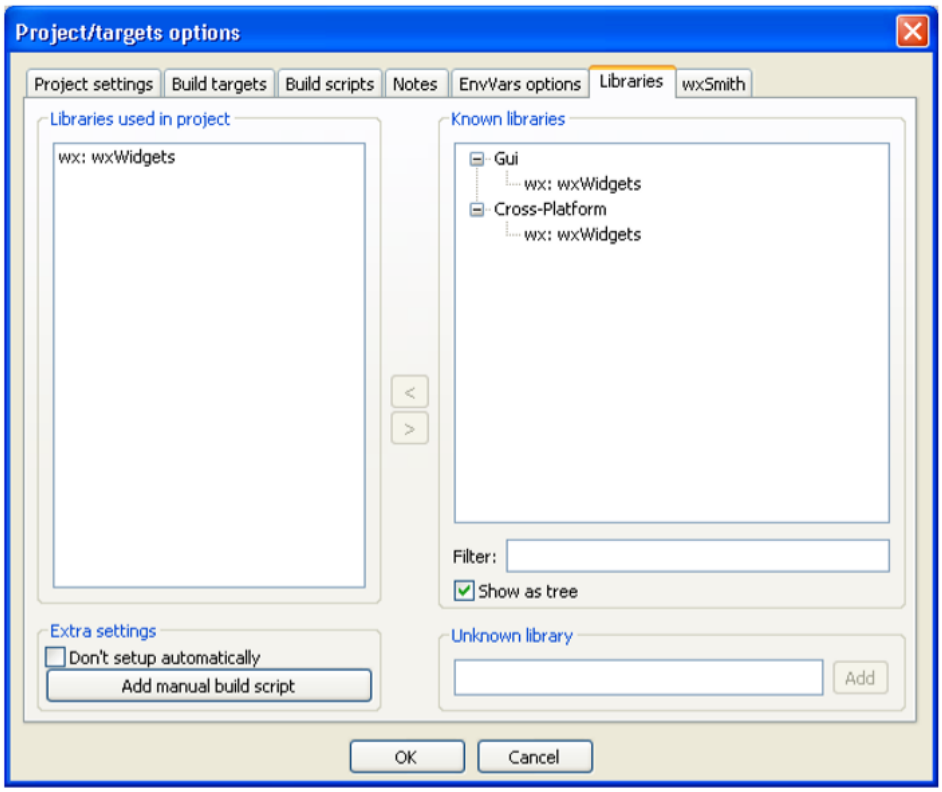


Рис. 2.12. Конфигурация проекта

Вы можете фильтровать библиотеки, известные LibFinder, с помощью фильтра поиска. Флажок 'Show as Tree' позволяет переключаться между видом по категориям и видом без рубрик.

Если вы хотите добавить библиотеку, которая не доступна в базе данных LibFinder, вы можете использовать поле 'Unknown Library'. Заметьте, что вы должны ввести короткий код (shortcode) библиотеки (который обычно совпадает с именем глобальной переменной) или имя библиотеки в pkg-config. Список предлагаемых shortcodes может быть найден в Global Variables. Использование этой опции рекомендуется только тогда, когда проект готовится для сборки на другом компьютере, где такая библиотека есть и благополучно обнаруживается LibFinder. Вы имеете доступ к глобальным переменным в CodeBlocks так:

```
$(#GLOBAL_VAR_NAME.include)
```

Установка флажка опции 'Don't setup automatically' будет указывать LibFinder , что не следует добавлять библиотеки автоматически при компиляции проекта. В этом случае LibFinder может вызываться из скрипта сборки. Пример подобного скрипта генерируется и добавляется к проекту при нажатии 'Add manual build script' .

2.10.3 Использование LibFinder и проектов из помощников (wizards)

Помощники создают проекты, не используя LibFinder. Чтобы интегрировать проект с этим плагином, вы должны вручную обновить опции сборки проекта. Это может быть легко сделано удалением всех специфических для библиотек установок и добавлением библиотек через закладку 'Libraries' в свойствах проекта.

Такой проект станет кросс-платформенным. Как только библиотеки определены в базе данных LibFinder, опции сборки проекта будут автоматически обновлены для соответствия специфическим библиотечным установкам платформы.

2.11 AutoVersioning

Плагин версий приложений обновляет версию и номер сборки вашего приложения каждый раз, как изменение было сделано и сохранено в version.h, с помощью лёгкого использования объявлений переменной. Также есть возможность передачи изменений в стиле SVN, редакторе схем версий, генератора изменений и т.п.

2.11.1 Введение

Идея плагина AutoVersioning появилась в процессе разработки pre-alpha программы, которая требовала информации о версии и состоянии. Будучи загружены кодированием, при отсутствии времени для слежения за номером версии мы решили разработать плагин, который будет выполнять эту работу с наименьшим возможным вмешательством.

2.11.2 Возможности

Вот список суммарных возможностей плагина:

- Поддержка C и C++.
- Генерация и автоматическое обновление переменных версии.
- Редактор состояния программы.
- Интегрированный редактор схемы изменений и поведения автоматически обновляемых переменных версии.
- Объявление даты, то есть, месяц, дата, год.
- Ubuntu стиль версии.
- Проверка ревизии Svn.
- Генератор журнала изменений.
- Работа в Windows и Linux.

2.11.3 Использование

Просто зайдите в меню 'Project' →'Autoversioning'. Появится всплывающее окно подобное этому:

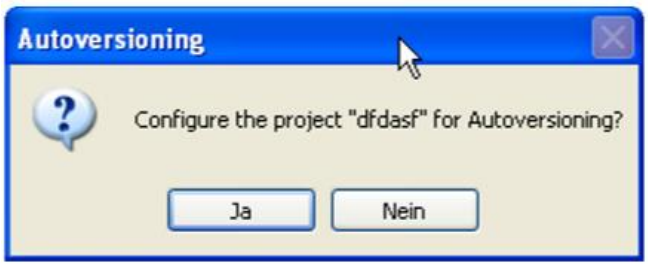


Рис. 2.13. Конфигурирование проекта для Autoversioning

При нажатии на **yes** в ответ на запрос сообщения откроется основной диалог конфигурации autoversioning, что позволит вам сконфигурировать информацию о версии вашего проекта.

После конфигурирования проекта для auto versioning, настройки, которые вы сделали в диалоге, будут храниться в файле проекта, а файл version.h будет создан. Теперь каждый раз, когда вы используете меню 'Project' →'Autoversioning', будет появляться диалог конфигурации, чтобы вы

могли отредактировать версию вашего проекта и относящиеся к версии настройки, пока вы не сохраните новые изменения с помощью плагина в файле проекта.

2.11.4 Закладки диалога блокнота

2.11.4.1 Значения версии

Здесь вы вводите только соответствующие значения версии или разрешаете плагину auto versioning увеличить их для вас (см. Рис. 2.14 на стр. 46).

Major Увеличивается на 1, когда младшая версия достигает своего максимума.

Minor Увеличивается на 1, когда номер сборки переходит барьер времени сборки, значение сбрасывается в 0, когда достигается максимальное значение.

Build Number (также эквивалентно Release) – увеличивается на 1 каждый раз, когда увеличивается номер ревизии.

Revision Увеличивается случайно, когда проект модифицируется, а затем компилируется.

2.11.4.2 Состояние

Некоторые поля сохраняют след состояния вашего проекта со списком предопределённых значений для удобства (см. рис. 2.15 на стр. 46).

Software Status Типичный пример этого: v1.0 Alpha.

Abbreviation Похоже на статус, но ближе к этому: v1.0a

2.11.4.3 Схема

Позволяет вам редактировать то, как плагин будет увеличивать значения версии (см. рис. 2.16 на стр. 47).

Minor maximum Максимальный номер, которого значение Minor может достигать; после того как это значение будет достигнуто, Major увеличивается на 1, а когда затем проект компилируется, Minor устанавливается в 0.

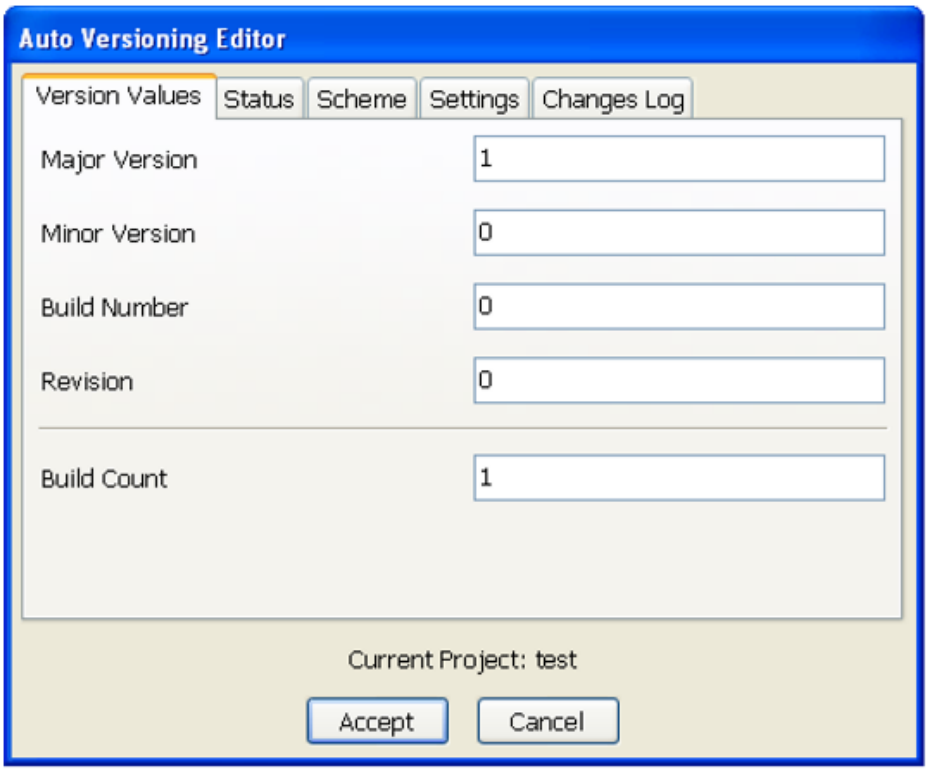


Рис. 2.14. Установка значений версии

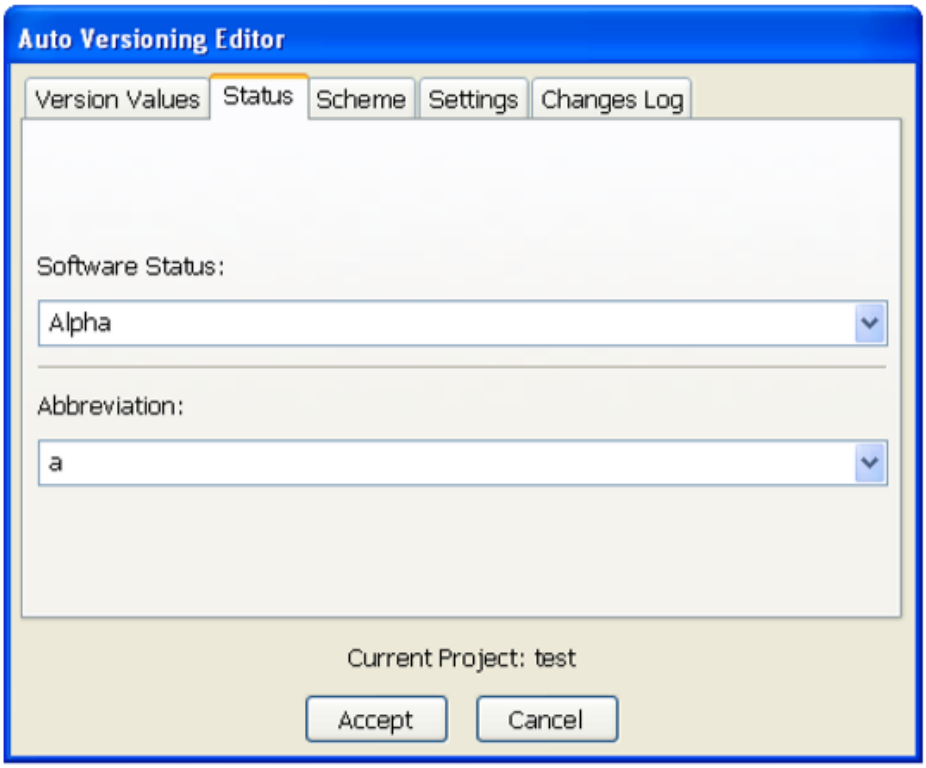


Рис. 2.15. Установка состояния Autoversioning

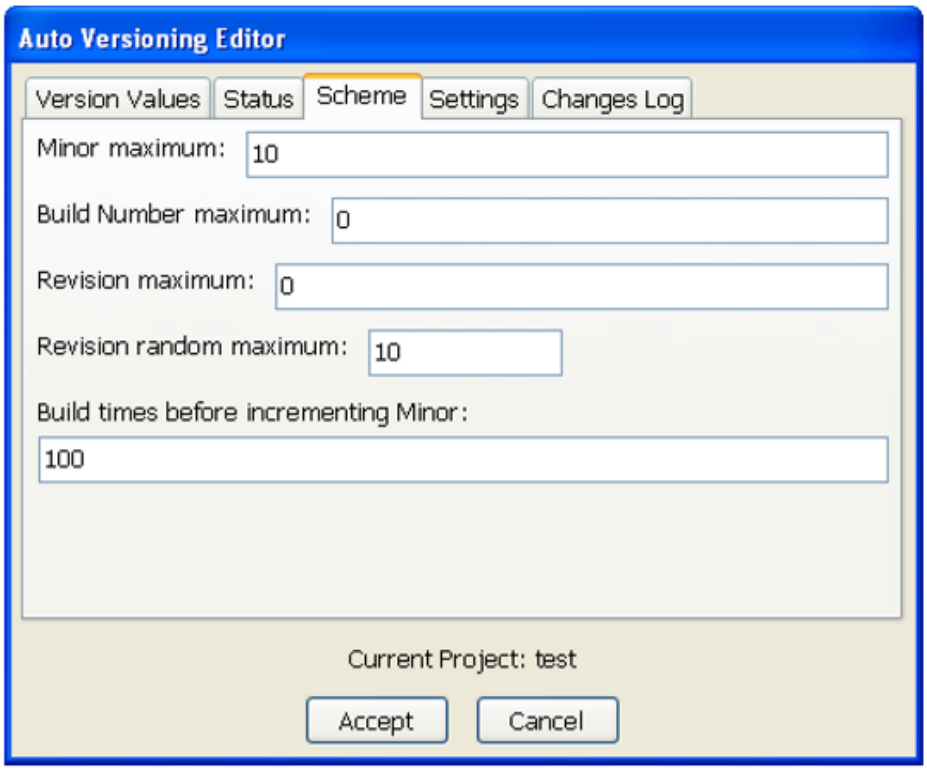


Рис. 2.16. Схема autoversioning

Build Number maximum Когда значение достигнуто, в следующий раз при компиляции проекта установится в 0. Чтобы снять ограничения впишите 0.

Revision maximum Почти как Build Number maximum. Установите в 0, чтобы снять ограничения.

Revision random maximum Ревизия увеличивается случайными числами, какими вы решите; если вы зададите здесь 1, ревизия будет увеличиваться на 1 безусловно.

Build times before incrementing Minor После успешных изменений в коде и компиляции история сборок будет увеличена, а когда достигнет этого значения, будет увеличено Minor.

2.11.4.4 Установки

Здесь вы можете задать некоторые настройки поведения auto versioning (см. рис. 2.17 на стр. 48).

Autoincrement Major and Minor Позволяет плагину увеличивать это значение, используя схему. Если не отмечено, только Build Number и Revision будут увеличиваться.

Create date declarations Создаёт ввод в файл version.h с датой и версией в Ubuntu стиле.

Do Auto Increment Этим сообщается плагину автоматически увеличивать изменения, когда происходит модификация; это увеличение будет обнаружено до компиляции.

Header language Выбирает язык вывода в version.h

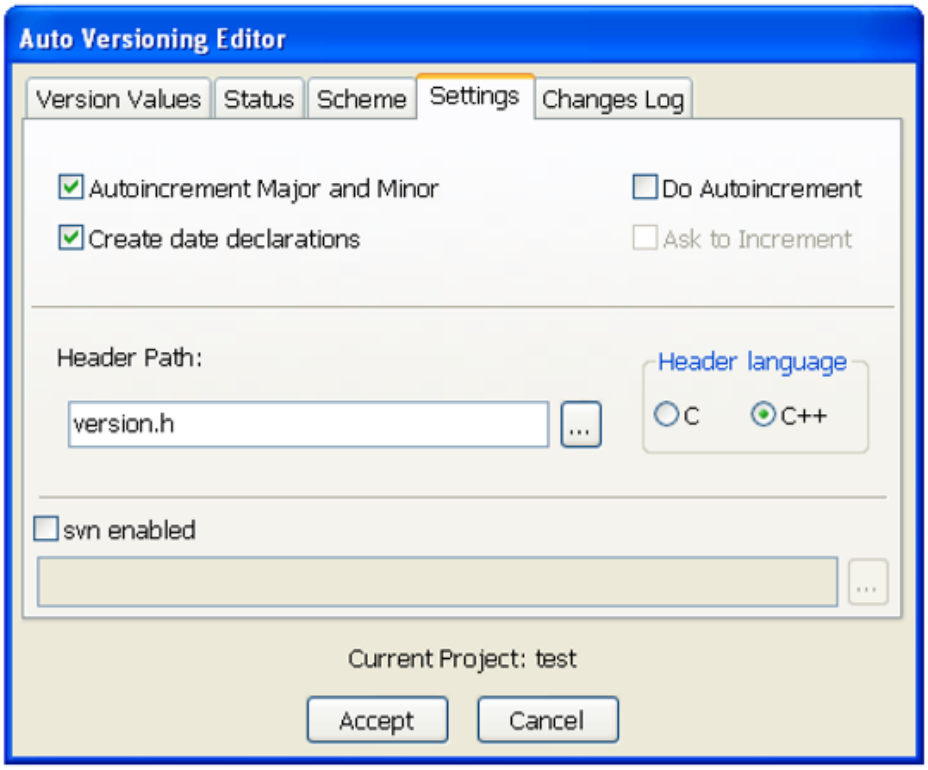


Рис. 2.17. Установки Autoversioning

Ask to increment Если отмечено Do Auto Increment, здесь вас спрашивают перед компиляцией (если изменения были сделаны), увеличивать ли значения версии.

svn enabled Ищется svn ревизия и дата в текущей папке и генерируется правильный ввод в version.h

2.11.4.5 Журнал изменений

Это позволяет вам вводить все сделанные изменения в проект для генерации ChangesLog.txt файла (см. рис. 2.18 на стр. 49).

Show changes editor when incrementing version Вызывает появление редактора журнала изменений, когда увеличивается версия.

Title Format Формат доступного заголовка со списком предопределённых значений.

2.11.5 Включение в ваш код

Для использования переменных, генерированных плагином, достаточно #include <version.h>. Пример кода будет похож на этот:

```
#include <iostream> #include "version.h"

void main(){ std::cout<<AutoVersion::Major<<endl; }
```

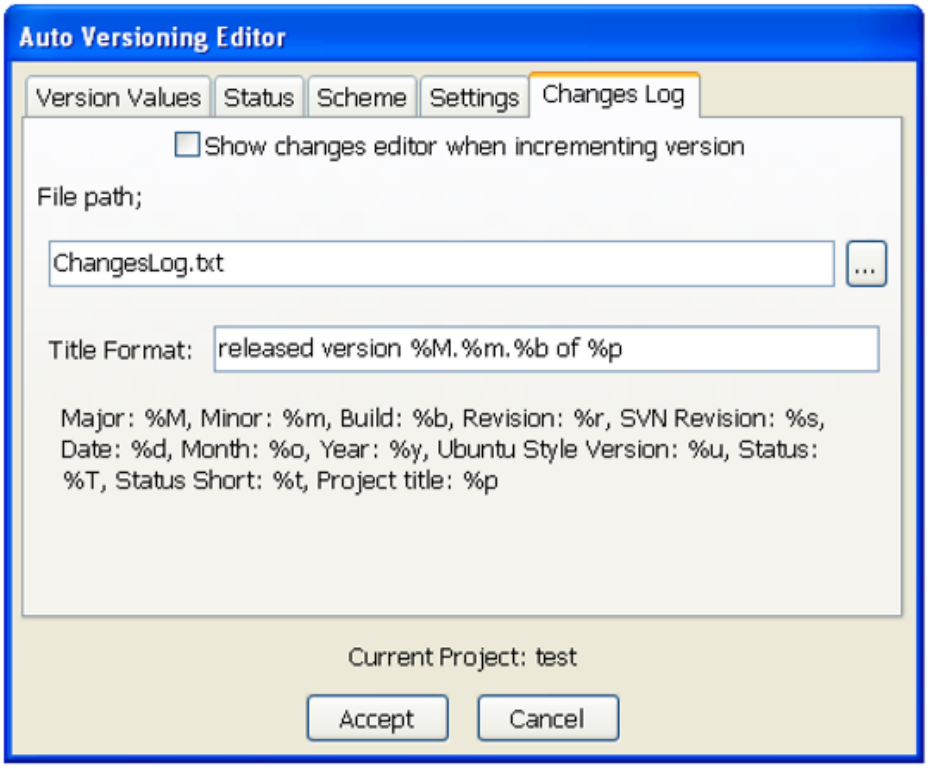


Рис. 2.18. Changelog в Autoversioning

2.11.5.1 Вывод version.h

Сгенерированный заголовочный файл. Вот пример содержания файла в режиме C++:

```
#ifndef VERSION_H #define VERSION_H

namespace AutoVersion{

//Date Version Types static const char DATE[] = "15"; static const char
MONTH[] = "09"; static const char YEAR[] = "2007"; static const double
UBUNTU_VERSION_STYLE = 7.09;

//Software Status static const char STATUS[] = "Pre-alpha"; static const char
STATUS_SHORT[] = "pa";

//Standard Version Type static const long MAJOR = 0; static const long MINOR
= 10; static const long BUILD = 1086; static const long REVISION = 6349;

//Miscellaneous Version Types static const long BUILDS_COUNT = 1984; #define
RC_FILEVERSION 0,10,1086,6349

#define RC_FILEVERSION_STRING "0, 10, 1086, 6349\0" static const char
FULLVERSION_STRING[] = "0.10.1086.6349";

} #endif //VERSION_h
```

В режиме Си он такой же, как в C++, но без namespace:

```
#ifndef VERSION_H #define VERSION_H
```

```
//Date Version Types static const char DATE[] = "15"; static const char
MONTH[] = "09"; static const char YEAR[] = "2007"; static const double
UBUNTU_VERSION_STYLE = 7.09;

//Software Status static const char STATUS[] = "Pre-alpha"; static const char
STATUS_SHORT[] = "pa";

//Standard Version Type static const long MAJOR = 0; static const long MINOR
= 10; static const long BUILD = 1086; static const long REVISION = 6349;

//Miscellaneous Version Types static const long BUILDS_COUNT = 1984; #define
RC_FILEVERSION 0,10,1086,6349 #define RC_FILEVERSION_STRING "0, 10, 1086,
6349\0" static const char FULLVERSION_STRING[] = "0.10.1086.6349";

#endif //VERSION_h
```

2.11.6 Генератор журнала изменений

Этот диалог доступен из меню 'Project' →'Changes Log'. Так же, если установлено 'Show changes editor when incrementing version' на странице установок changes log, окно будет открыто, чтобы позволить вам ввести список изменений после модификации в исходниках проекта, или когда увеличение имело место (см. рис. 2.19 на стр. 50).

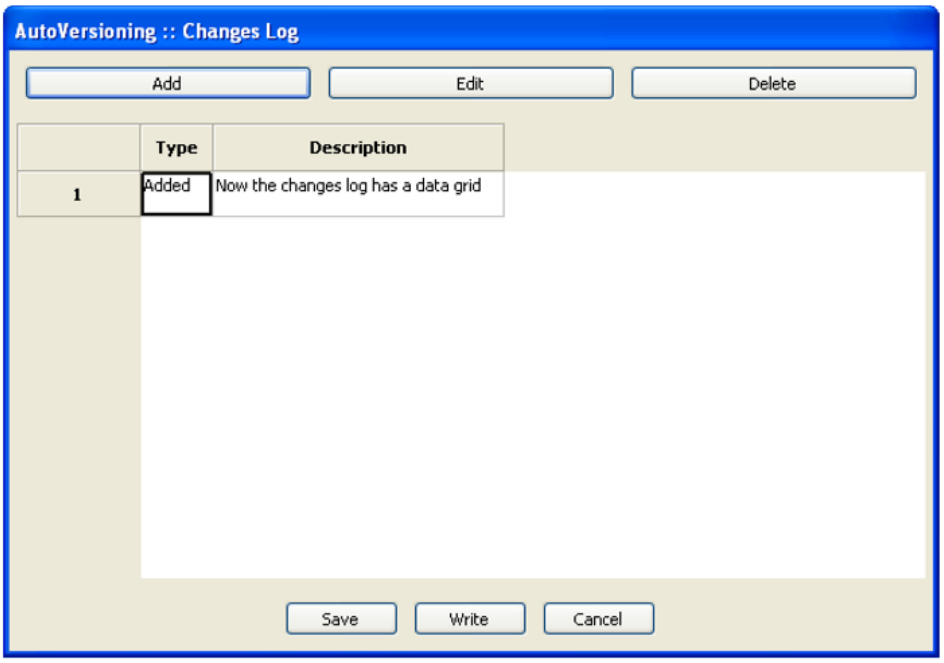


Рис. 2.19. Изменения в проекте

2.11.6.1 Кнопки суммарно

Add Присоединяет строку к сетке данных.

Edit Разрешает модификацию выбранной ячейки.

Delete Удаляет текущую строку из сетки данных.

Save Сохраняет во временном файле (changes.tmp) текущие данные для последнего процесса в файле журнала изменений.

Write Передаёт данные сетки данных в файл журнала изменений.

Cancel Только закрывает диалог без выполнения каких-либо действий.

Вот пример вывода, сгенерированного плагином в файл ChangesLog.txt:

```
03 September 2007 released version 0.7.34 of AutoVersioning-Linux

Change log: -Fixed: pointer declaration -Bug: blah blah

02 September 2007 released version 0.7.32 of AutoVersioning-Linux

Change log: -Documented some areas of the code -Reorganized the code for
readability

01 September 2007 released version 0.7.30 of AutoVersioning-Linux

Change log: -Edited the change log window -If the change log windows is leave
blank no changes.txt is modified
```

2.12 Code statistics

Рис. 2.20. Конфигурация статистики кода

Основанный на вводе в маску конфигурации этот простой плагин обнаруживает размеры кода, комментарии и пустые строки проекта. Расчёты вызываются командой меню 'Plugins' →'Code statistics'.

2.13 Searching Available Source Code

Этот плагин делает возможным выбрать выражение в редакторе и искать это выражение через контекстное меню 'Search at Koders' в [,→Koders] базе данных. Диалог даёт дополнительные возможности для фильтрации языков программирования и лицензий.

Этот поиск в базе данных поможет вам найти исходный код в других проектах университетов, консорциумов и таких организаций, как Apache, Mozilla, Novell Forge, SourceForge и многих других, код, который может быть использован без необходимости каждый раз изобретать колесо. Пожалуйста, обратите внимание на лицензию исходного кода в каждом индивидуальном случае.

2.14 Code profiler

Простой графический интерфейс к GNU GProf Profiler.

2.15 Symbol Table Plugin

Этот плагин делает возможным искать символы в объектных файлах и библиотеках. Опции и пути для программы в командной строке nm определены на закладке Options.

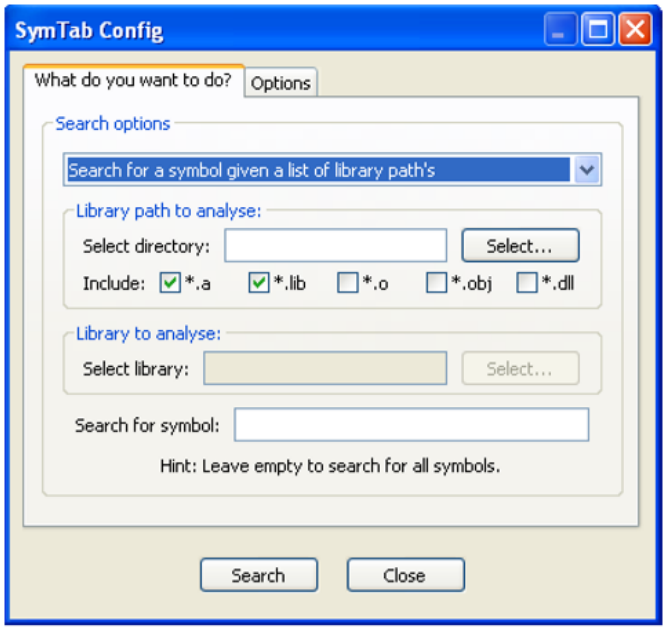


Рис. 2.21. Конфигурирование Symbol Table

Щелчок по **Search** начнёт поиск, результаты NM программы отобразятся в отдельном окне, названном 'SymTabs Result'. Имя объектного файла или библиотеки, содержащей символ, будут перечислены под заголовком 'NM's Output'.

3 Расширение переменных

CodeBlocks делает различия между несколькими типами переменных. Эти типы служат целям конфигурации окружения при создании программ, и в то же время улучшают надёжность и переносимость. Доступ к переменным CodeBlocks выполняется через $<name>.

Envrionment Variable устанавливаются при запуске CodeBlocks. Они могут модифицировать системные переменные окружения, такие как PATH. Это может быть полезно в тех случаях, когда определённое окружение необходимо для создания проектов. Установки для переменных окружения в CodeBlocks создаются в 'Settings' →'Environment' →'Environment Variables'.

Builtin Variables предопределены в CodeBlocks, и могут быть доступны через их имена (см. раздел 3.2 на стр. 54 для уточнения).

Command Macros Этот тип переменных используется для управления процессом сборки. За уточнениями обратитесь к разделу 3.4 на стр. 58.

Custom Variables определённые пользователем переменные, которые могут быть заданы в опциях сборки проекта. Здесь вы можете, например, определить ваши производные как переменные MCU и присвоить им соответствующие значения. Затем установить опцию компилятора -mcpu=$(MCU), а CodeBlocks автоматически заменит содержимое. Этим методом установки для проекта могут быть позже параметризованы.

Global Variables в основном используются для создания CodeBlocks из исходников или разработок wxWidgets приложений. Эти переменные имеют очень специальное значение. В отличие от всех других, если вы задаёте такие переменные и передаёте файл вашего проекта другим, у которых *not* в установках этих GV, CodeBlocks будет запрашивать у пользователя установку переменных. Это очень лёгкий путь убедить «других разработчиков», что задать легче. CodeBlocks будет запрашивать все обычно необходимые пути.

3.1 Синтаксис

CodeBlocks обрабатывает следующие функционально идентичные последовательности символов в pre-build, post-build или во время сборки как переменные:

- $VARIABLE
- $(VARIABLE)
- ${VARIABLE}
- %VARIABLE%

Имена переменных должны состоять из буквенно-цифровых символов, и они не чувствительны к регистру. Переменные, начинающиеся с отдельного символа решётки (#) интерпретируются как глобальные пользовательские переменные (см. раздел 3.7 на стр. 58 для уточнения). Имена, перечисленные ниже, интерпретируются как встроенные типы.

Переменные, которые не являются ни глобальными пользовательскими, ни встроенными типами будут замещены значениями предусмотренными в файле проекта или переменными среды, если последние подведут.

> Примечание:
>
> Целевые определения предшествуют определениям проекта.

3.2 Список доступных встроенных переменных

Переменные, перечисленные здесь, это встроенные переменные CodeBlocks. Они не могут использоваться в файлах исходного кода.

3.2.1 Рабочее пространство CodeBlocks

`$(WORKSPACE_FILENAME), $(WORKSPACE_FILE_NAME), $(WORKSPACEFILE), $(WORKSPACEFILENAME)`

Имя файла текущего рабочего пространства проекта (.workspace).

`$(WORKSPACENAME), $(WORKSPACE_NAME)`

Имя рабочего пространства, которое отображается на закладке Projects панели Management.

`$(WORKSPACE_DIR), $(WORKSPACE_DIRECTORY), $(WORKSPACEDIR), $(WORKSPACEDIRECTORY)`

Расположение директории рабочего пространства.

3.2.2 Файлы и директории

`$(PROJECT_FILENAME), $(PROJECT_FILE_NAME), $(PROJECT_FILE), $(PROJECTFILE)`

Имя файла текущего скомпилированного проекта.

`$(PROJECT_NAME)`

Имя текущего скомпилированного проекта.

`$(PROJECT_DIR), $(PROJECTDIR), $(PROJECT_DIRECTORY)`

Общая директория верхнего уровня текущего скомпилированного проекта.

`$(ACTIVE_EDITOR_FILENAME)`

Имя файла, открытого в настоящий момент в активном редакторе.

`$(ACTIVE_EDITOR_LINE)`

Возвращает текущую строку в активном редакторе.

`$(ACTIVE_EDITOR_COLUMN)`

Возвращает колонку в текущей строке активного редактора.

`$(ACTIVE_EDITOR_DIRNAME)`

Директория, содержащая текущий активный файл (относительно общего пути верхнего уровня).

`$(ACTIVE_EDITOR_STEM)`

Базовое имя (без расширения) текущего активного файла.

`$(ACTIVE_EDITOR_EXT)`

Расширение текущего активного файла.

`$(ALL_PROJECT_FILES)`

Строка, содержащая имена всех файлов в текущем проекте.

`$(MAKEFILE)`
: Имя файла makefile.

`$(CODEBLOCKS), $(APP_PATH), $(APPPATH), $(APP-PATH)`
: Путь к текущему работающему образцу CodeBlocks.

`$(DATAPATH), $(DATA_PATH), $(DATA-PATH)`
: «Общая» директория работающего текущего образца CodeBlocks.

`$(PLUGINS)`
: Директория плагинов текущего работающего образца CodeBlocks.

`$(TARGET_COMPILER_DIR)`
: Директория установки компилятора, так называемая мастер path.

3.2.3 Цели сборки

`$(FOOBAR_OUTPUT_FILE)`
: Выходной файл заданной цели.

`$(FOOBAR_OUTPUT_DIR)`
: Выходная директория заданной цели.

`$(FOOBAR_OUTPUT_BASENAME)`
: Базовое имя выходного файла (без пути, без расширения) заданной цели.

`$(TARGET_OUTPUT_DIR)`
: Выходная директория текущей цели.

`$(TARGET_OBJECT_DIR)`
: Директория объектных файлов текущей цели.

`$(TARGET_NAME)`
: Имя текущей цели.

`$(TARGET_OUTPUT_FILE)`
: Выходной файл текущей цели.

`$(TARGET_OUTPUT_BASENAME)`
: Базовое имя выходного файла (без пути, без расширения) текущей цели.

`$(TARGET_CC), $(TARGET_CPP), $(TARGET_LD), $(TARGET_LIB)`
: Исполняемые файлы инструментов сборки (компилятор, компоновщик и т.д.) текущей цели.

3.2.4 Язык и кодировка

`$(LANGUAGE)`
: Системный язык в открытом тексте.

`$(ENCODING)`
: Кодировка символов в открытом тексте.

3.2.5 Время и дата

`$(TDAY)`
: Текущая дата в форме YYYYMMDD (например, 20051228).

`$(TODAY)`

Текущая дата в форме YYYY-MM-DD (например, 2005-12-28).

`$(NOW)`

Время создания файла в форме YYYY-MM-DD-hh.mm (например, 2005-1228-07.15).

`$(NOW_L)`

Время создания/модификации файла в форме YYYY-MM-DD-hh.mm.ss (например, 200512-28-07.15.45)

`$(WEEKDAY)`

День недели в простом тексте (например, 'Wednesday').

`$(TDAY_UTC)`, `$(TODAY_UTC)`, `$(NOW_UTC)`, `$(NOW_L_UTC)`, `$(WEEKDAY_UTC)`

Идентично предыдущим типам, но расширено относительно UTC.

`$(DAYCOUNT)`

Количество дней, прошедших с произвольно выбранного нулевого дня (January 1, 2009). Полезно в качестве последнего компонента номера версии/сборки.

3.2.6 Случайные значения

`$(COIN)`

Эта переменная «бросает виртуальные монетки» (по одной на вызов) и возвращает 0 или 1.

`$(RANDOM)`

16-битовое положительное случайное число (0-65535).

3.2.7 Команды операционной системы

Эти переменные замещают команды операционной системы.

`$(CMD_CP)`

Команда копирования файлов.

`$(CMD_RM)`

Команда удаления файлов.

`$(CMD_MV)`

Команда перемещения файлов.

`$(CMD_MKDIR)`

Команда создания директории.

`$(CMD_RMDIR)`

Команда удаления директории.

3.2.8 Оценка условий

```
$if(condition){true clause}{false clause}
```

Оценка условий будет сделана как true (истина), если

- Условие не пустая последовательность символов отличная от 0 или false.
- Условие не пустая переменная, которая отлична от 0 или false.

- Условие – это переменная, которая вычисляется как true (подразумеваемое предыдущим условием).

Оценка условия будет сделана как false (ложь), если

- Условие пусто.
- Условие 0 или false.
- Условие – это переменная, которая пуста или вычисляется как 0 или false.

> Примечание:
>
> Пожалуйста, отметьте, что ни варианты синтаксиса переменной %if (...), ни $(if)(...) не поддерживаются этой конструкцией.

Пример

Например, если вы используете несколько платформ, и вы хотите задать разные параметры, зависящие от операционной системы. В следующем коде команды скрипта [[ ]] определяются, и <command> будет выполняться. Это может быть полезно на post-built этапе.

```
[[ if (PLATFORM == PLATFORM_MSW) { print (_T("cmd /c")); } else { print
(_T("sh ")); } ]] <command>
```

3.3 Расширение скрипта

Для максимальной гибкости вы можете встроить скрипты, используя оператор [[ ]] в качестве специального случая расширения переменной. Встроенные скрипты имеют доступ ко всей доступной функциональности скриптов и прекрасно работают, во многом напоминая bash backticks, обратные галочки (исключая доступ к рабочему пространству CodeBlocks). Таким образом, скрипты не ограничиваются выводом текста, но могут также манипулировать состоянием (проекты, цели и т.д.).

> Примечание:
>
> Манипуляции состоянием CodeBlocks должны быть реализованы скорее предваряющим сборку (pre-build) скриптом, а не просто скриптом.

Пример с Backticks

```
objdump -D `find . -name *.elf` > name.dis
```

Выражения в «обратных галочках, backticks» возвращает список всех исполняемых *.elf в любых поддиректориях. Результат этого выражения может быть использован непосредственно objdump. Окончательный вывод передаётся в файл, названный name.dis. Так процесс может быть автоматизирован простым путём без добавления в программу каких-либо циклов.

Пример использования скрипта

Текст скрипта замещается любым выводом, сгенерированным вашим скриптом или отброшен в случае синтаксической ошибки.

Поскольку оценка условий работает до выполнения скриптов, оценка условий может быть использована для работы препроцессора. Встроенные переменные (и переменные пользователя) расширяются после скриптов, так что возможны ссылки на переменные на выходе скрипта.

```
[[ print(GetProjectManager().GetActiveProject().GetTitle()); ]]
```

вставляет заголовок активного проекта в командную строку.

3.4 Команды макросов

`$compiler`	Доступ к имени исполняемого файла компилятора.
`$linker`	Доступ к имени исполняемого файла компоновщика.
`$options`	Флаги компилятора.
`$link_options`	Флаги компоновщика.
`$includes`	Пути include компилятора.
`$c`	Пути include компоновщика.
`$libs`	Библиотеки компоновщика.
`$file`	Исходный файл (полное имя).
`$file_dir`	Директория исходного файла без имени файла и его расширения.
`$file_name`	Имя исходного файла без информации о пути и расширения имени файла.
`$exe_dir`	Директория исполняемого файла без имени файла и его расширения.
`$exe_name`	Имя исполняемого файла без пути и расширения имени файла.
`$exe_ext`	Расширение имени исполняемого файла без пути и имени файла.
`$object`	Объектный файл.
`$exe_output`	Исполняемый выходной файл.
`$objects_output_dir`	Выходная директория объектных файлов.

3.5 Компиляция единственного файла

```
$compiler $options $includes -c $file -o $object
```

3.6 Компоновка объектных файлов с исполняемым

```
$linker $libdirs -o $exe_output $link_objects $link_resobjects $link_options
$libs
```

3.7 Глобальные переменные компилятора

3.8 Резюме

Работа над проектом, который доверяется библиотекам 3й стороны, вовлекает во множество ненужных повторяющихся задач, таких, как задание переменных сборки, соответствующих виду локальной файловой системы. В случае файлов проекта следует озаботиться, чтобы избежать случайного появления локально модифицированных копий. Если не уделять этому внимания, это может легко вызвать, например, последующее изменение флажка сборки при создании окончательной сборки.

Концепция глобальных переменных компилятора – это уникальное новое решение CodeBlocks, которое адресовано этой проблеме. Глобальные переменные компилятора позволяют вам задать всё для проекта один раз при любом количестве разработчиков, использующих любое количество разных файловых систем, которые имеют возможность компилировать и разрабатывать этот проект. Никакой информации о локальном виде нет необходимости менять более одного раза.

3.9 Имена и члены

Глобальные переменные компилятора в CodeBlocks отделены от переменных проекта предваряющим значком решётки (диеза). Глобальные переменные компилятора структурированы; каждая переменная состоит из имени и опционно членства. Имена определяются свободно, тогда как некоторые члены встроены в IDE. Хотя вы можете в принципе выбрать любое имя переменной, желательно подобрать известный идентификатор для общих пакетов. Так что, количество информации, требуемой для поддержки пользователя, будет минимизировано. Группа CodeBlocks предоставляет список рекомендованных переменных для известных пакетов.

Членская база разрешает те же значения, что и имена переменных, используемых без членства (alias).

Включённые члены и библиотеки по умолчанию представлены с условными именами (aliases) для базы/включение и базы/библиотеки, соответственно. Однако пользователь может переопределить их, если нужны другие установки.

Обычно рекомендуют использовать синтаксис $(#variable.include) вместо $(#variable)/include, что придаёт больше гибкости, а вместе с тем идентично по функциональности (см. подраздел 3.12.1 на стр. 62 и рис. 3.1 на стр. 60 для уточнения).

Членские cflags и lflags пусты по умолчанию и могут использоваться для поддержки возможности дать тот же совместный набор флагов compiler/linker для всех сборок на одной машине. CodeBlocks позволяет вам определять пользовательские членские переменные вдобавок к встроенным.

3.10 Ограничения

- И заданные, и глобальные имена переменных компилятора не могут быть пустыми, они не должны содержать пробелов, должны начинаться с буквы и должны состоять из буквенно-цифровых символов. Cyrillic или Chinese буквы не являются буквенно-цифровыми символами. Если CodeBlocks получает неверную символьную последовательность в качестве имени, программа может заменить её, не спрашивая.
- Каждая переменная требует, чтобы её база была определена. Всё остальное не обязательно, но база абсолютно обязательна. Если вы не определили базу переменной, она не будет сохраняться (неважно, какие другие поля вы определили).
- Вы не можете определять пользовательское членство, имеющее то же имя, что и встроенный член. В настоящее время пользовательские члены будут переписаны встроенными, но, в основном, поведение в этом случае не предсказуемо.
- Переменные и значения членов могут содержать произвольные символьные последовательности, с учётом следующих трёх ограничений:

- Вы не можете определять переменную значением, которое ссылается на ту же переменную или любого её члена.

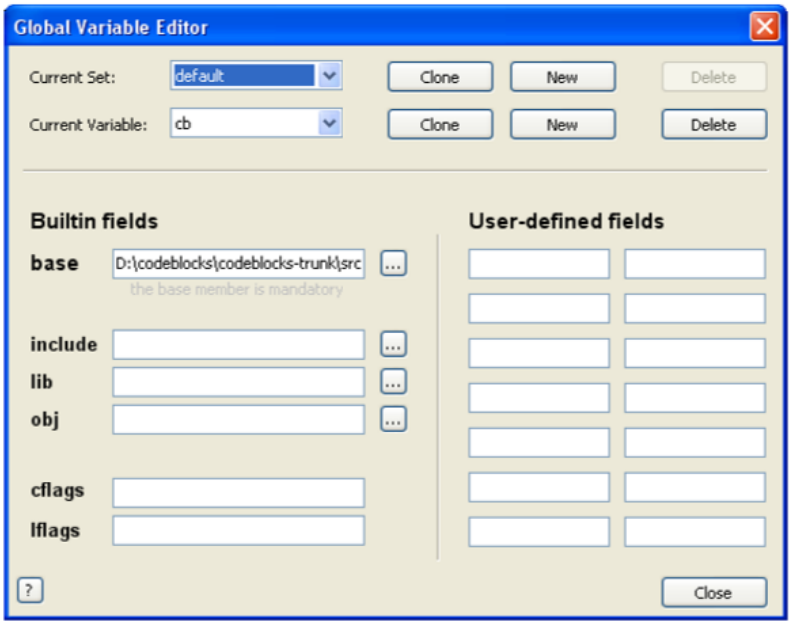


Рис. 3.1. Глобальные переменные окружения

- Вы не можете определять члена значением, которое ссылается на него же.
- Вы не можете определять члена или переменную значением, которое ссылается на ту же переменную или член через циклическую зависимость.

CodeBlocks обнаруживает наиболее явные случаи рекурсивных определений (которые могут возникнуть по оплошности), но не будет выполнять углублённый анализ каждого возможного злоупотребления. Если вы ввели чушь, тогда чушь – это то, что вы получите. Вы предупреждены.

Примеры

Определение wx.include как $(#wx)/include – это лишнее, но вполне допустимо. Определение wx.include как $(#wx.include) – это недопустимо, и будет обнаружено CodeBlocks. Определение wx.include как $(#cb.lib), которое вновь определено как $(#wx.include), создаст бесконечный цикл.

3.11 Использование глобальных переменных компилятора

Всё, что вам необходимо сделать для использования глобальных переменных компилятора, это положить их в ваш проект! Да, это так легко.

Когда IDE обнаруживает присутствие неизвестной глобальной переменной, вы получите запрос – ввести её значение. Значение будет храниться в ваших настройках, так что вам не будет нужды вводить информацию дважды.

Если вам нужно изменить или удалить переменную позже, вы может это сделать в меню установок.

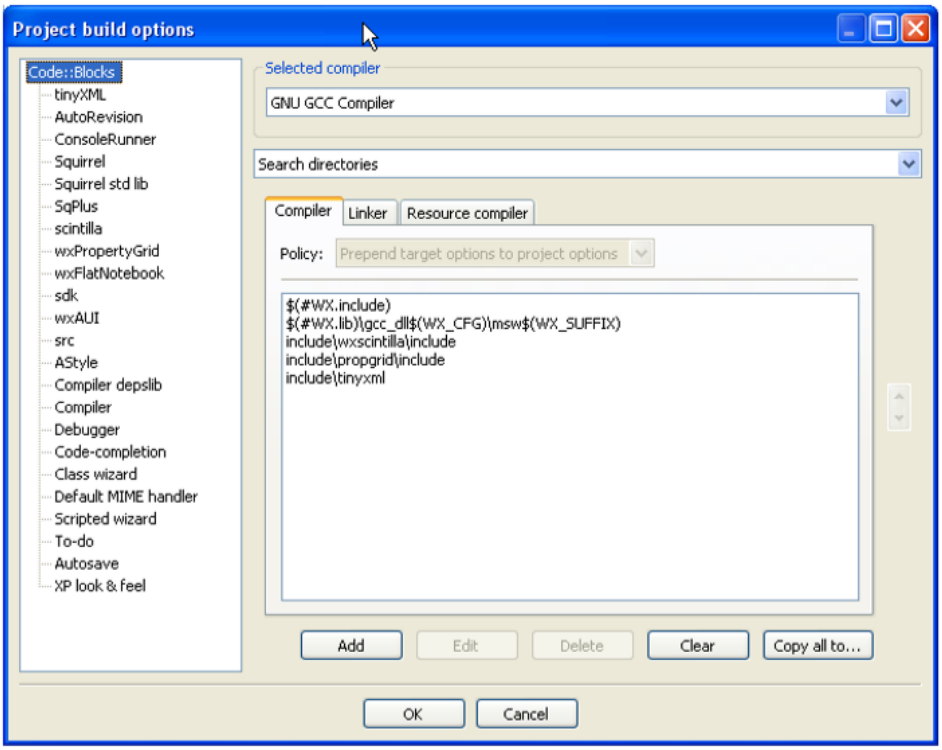


Рис. 3.2. Глобальные переменные

Пример

Картинка выше показывает и проект, и глобальные переменные. WX_SUFFIX определено в проекте, но WX – это глобальная пользовательская переменная.

3.12 Набор переменных

Иногда вы хотите использовать разные версии той же библиотеки или вы разрабатываете две ветви той же программы. Хотя допустимо продолжать с глобальной переменной компилятора, это может стать довольно утомительно. Для этого случая CodeBlocks поддерживает набор переменных. Набор переменных – это независимая коллекция переменных, идентифицированных по имени (набор имён имеет те же ограничения, что и имена переменных).

Если вы хотите переключаться между разными наборами переменных, вы просто должны выбрать разные наборы из меню. Разные наборы не требуют, чтобы были те же переменные, а идентичные переменные в разных наборах не обязаны иметь одинаковые значения или даже тех же пользовательских членов.

Другим положительным свойством наборов является то, что если вы имеете дюжину переменных, а вы хотите иметь новый набор с одной из этих переменных, размещённых в разных местах, вам не требуется заново вводить все данные. Вы можете просто создать клон вашего текущего набора, который продублирует все ваши переменные.

Удаление набора удаляет также все переменные в этом наборе (но не в другом наборе). Предопределённый набор всегда присутствует и не может быть удалён.

3.12.1 Пользовательские члены, мини-руководство

Как указано выше, записи $(#var.include) и $(#var)/include являются совершенно одинаковыми по умолчанию. Так что, отчего бы вам нужно было писать что-то интуитивно непонятное, как $(#var.include)?

Давайте возьмём стандартную установку Boost в Windows в качестве примера. Обычно вы ожидаете функционала пакета ACME, чтобы иметь включённые в него файлы под ACME/include и его библиотеки под ACME/lib. Дополнительно может иметь место наличие его заголовочных файлов в совсем другой поддиректории, названной acme. Так что после добавления правильного пути к опциям компилятора и компоновщика, вам потребуется #include <acme/acme.h> и связка с libacme.a (или какой бы она ни была).

URL каталог

[,→7Z] 7z zip home page.
http://www.7-zip.org
[,→BERLIOS] Codeblocks at berlios.
http://developer.berlios.de/projects/codeblocks/
[,→FORUM] Codeblocks forum.
http://forums.codeblocks.org/
[,→WIKI] Codeblocks wiki.
http://wiki.codeblocks.org/index.php?title=Main_Page/
[,→CODEBLOCKS] Codeblocks home page.
http://www.codeblocks.org/
[,→GCC] GCC home page.
http://gcc.gnu.org/
[,→HIGHTEC] HighTec home page.
http://www.hightec-rt.com/
[,→Koders] Koders home page.
http://www.koders.com/
[,→TriCore] TriCore home page.
http://www.infineon.com/tricore/
[,→TortoiseSVN] TriCore home page.
http://tortoisesvn.net/
[,→Subversion] TriCore home page.
http://subversion.tigris.org/
[,→Wxwidgets] WxWidgets home page.
http://www.wxwidgets.org/
[,→Wxcode] WxCode home page.
http://wxcode.sourceforge.net/
[,→Scripts] Scripting commands.
http://wiki.codeblocks.org/index.php?title=Scripting_commands/

Оглавление

1 Управление проектом 2
1.1 Обозреватель проекта 3
1.2 Примечания к проекту 4
1.3 Шаблоны проекта 4
1.4 Создание проектов из целей сборки 4
1.5 Виртуальные цели 4
1.6 Пред- и пост-сборочные шаги 5
Пример 5
1.7 Добавление скриптов в цель сборки 6
1.8 Рабочее пространство и зависимости проекта 6
Пример 6
1.9 Включение ассемблерных файлов 6
1.10 Редактор и инструменты 7
1.10.1 Предопределённый код 7
Пример 7
1.10.2 Аббревиатура 7
1.10.3 Персонализация 8
1.10.4 Файлы конфигурации 8
1.10.5 Навигация и поиск 9
1.10.6 Обозреватель символов 11
1.10.7 Включение внешних файлов помощи 13
1.10.8 Включение внешних инструментов 14
1.11 Краткие указания по работе с CodeBlocks 14
1.11.1 Отслеживание модификаций 14
1.11.2 Обмен данными с другими приложениями 15
1.11.3 Конфигурирование переменных окружения 15
Пример 16
1.11.4 Переключение между видами 17
1.11.5 Переключение между проектами 17
1.11.6 Расширенные установки для компиляторов 17
1.11.7 Масштабирование в редакторе 18
1.11.8 Режим оборачивания 18
1.11.9 Выбор режимов в редакторе 19
1.11.10 Свёртывание кода 19

1.11.11 Автоматическое завершение 20
1.11.12 Поиск повреждённых файлов 20
1.11.13 Включённые библиотеки 20
Пример 20
1.11.14 Порядок подключения объектных файлов 21
1.11.15 Автосохранение 21
1.11.16 Установки для расширений файлов 21
1.12 CodeBlocks в командной строке 21
1.13 Горячие клавиши 22
1.13.1 Editor 23
1.13.2 Files 23
1.13.3 View 23
1.13.4 Search 24
1.13.5 Build 24
2 Плагины 25
2.1 Astyle 25
2.2 CodeSnippets 26
2.3 Incremental Search 28
2.4 ToDo List 30
2.5 Source Code Exporter 31
2.6 Thread Search 31
2.6.1 Возможности 32
2.6.2 Использование 32
2.6.3 Конфигурация 32
2.6.4 Опции 33
2.6.5 Thread Search опции 34
2.6.6 Внешний вид 34
2.6.7 Панель Management 35
2.6.8 Регистратор типа 35
2.6.9 Режим разделения окон 35
2.6.10 Результаты краткого поиска 35
2.7 FileManager и PowerShell плагин 35
Пример 38
2.8 Browse Tracker 39

2.9 SVN поддержка 40
2.10 LibFinder 40
2.10.1 Поиск библиотек 41
2.10.2 Включение библиотек в проекты 42
2.10.3 Использование LibFinder и проектов из помощников (wizards) 43
2.11 AutoVersioning 44
2.11.1 Введение 44
2.11.2 Возможности 44
2.11.3 Использование 44
2.11.4 Закладки диалога блокнота 45
2.11.5 Включение в ваш код 48
2.11.6 Генератор журнала изменений 50
2.12 Code statistics 51
2.13 Searching Available Source Code 51
2.14 Code profiler 52
2.15 Symbol Table Plugin 52
3 Расширение переменных 53
3.1 Синтаксис 53
3.2 Список доступных встроенных переменных 54
3.2.1 Рабочее пространство CodeBlocks 54
3.2.2 Файлы и директории 54
3.2.3 Цели сборки 55
3.2.4 Язык и кодировка 55
3.2.5 Время и дата 55
3.2.6 Случайные значения 56
3.2.7 Команды операционной системы 56
3.2.8 Оценка условий 56
Пример 57
3.3 Расширение скрипта 57
Пример с Backticks 57
Пример использования скрипта 57
3.4 Команды макросов 58
3.5 Компиляция единственного файла 58
3.6 Компоновка объектных файлов с исполняемым 58

3.7 Глобальные переменные компилятора 58
3.8 Резюме 58
3.9 Имена и члены....59
3.10 Ограничения 59
Примеры....60
3.11 Использование глобальных переменных компилятора 60
Пример 61
3.12 Набор переменных....61
3.12.1 Пользовательские члены, мини-руководство 62
URL каталог....62